TECHNICAL REFERENCE MANUAL

NVIDIA TEGRA 4
4-PLUS-1 Quad-Core Processors

NVIDIA® Tegra® Processors: T40SW, T40X, T40S, AP40

Abstract

The Technical Reference Manual focuses on the logical structure and control of Tegra 4 (“Wayne” series) processors. It provides information for those modules that interface to external devices, or those that control fundamental chip operations. The modules detailed in this document provide an overview, any necessary programming guidelines, and a register listing for that module. Internal functional units such as video and graphics hardware acceleration are controlled by NVIDIA provided software and not documented here.

Revision History

Version	Date	Description
v01p	SEP 6, 2013	Public Release to support Open Source Development. The following section is undergoing further review and should be considered preliminary: • MIPI-DSI (Display Serial Interface)

Table of Contents

[THIS PAGE INTENTIONALLY LEFT BLANK]

1.0 INTRODUCTION

The NVIDIA® Tegra® 4 series processor is a complete applications and digital media system built around several powerful hardware elements:

- CPU Complex: Quad Cortex™-A15 Symmetric Multi-Processing ARM® Cores in a 4-PLUS-1™ configuration with a quad-core fast CPU complex and a fifth Battery Saver Core. The Cortex-A15 core features triple instruction issue and both out-of-order and speculative execution. It has full cache coherency support for the quad symmetric processors. All processors have 32 KB Instruction and 32 KB Data Level 1 caches; and there is a 2 MB shared Level 2 cache for the quad-core complex and a 512 KB Level 2 cache for the fifth core. The NVIDIA 4-PLUS-1 architecture uses the fifth Battery Saver Core, which operates exclusively with the main CPU complex, for very low-power, low-leakage operation at the light CPU loads common to multimedia and lightly loaded use situations.
- Memory Controller: dual-channel (2x 32-bit) DRAM interface providing more than twice the available bandwidth of Tegra 3 devices. LP-DDR2, LP-DDR3 and DDR3 DRAM types are all supported.
- Graphics: Ultra Low-power, high-performance NVIDIA GeForce® Graphics Processing Unit (GPU). Handles 2D graphics rendering and 3D pixel and vertex shading. Quad pixel rendering pipes are capable of fully using the enhanced bandwidth from the dual channel memory controller.
- Audio/Video Decoder: the Audio-Video Processor (AVP) subsystem includes dedicated audio and video decode hardware acceleration, an ARM7 processor, and embedded RAM. This subsystem provides full motion playback of up to 1440P high-definition video and supports H.264 BP/MP/HP/MVC, VC-1, VP8, MPEG-2 and MPEG-4 video standards and multiple audio standards with dedicated hardware.
- Video Encoder: A high performance H.264 1440P capable hardware video encoder. This processor supports H.264 BP/MP/HP/MVC and VP8 encoding.
- Imaging: A high-quality hardware accelerated still-image and video capture path supporting Bayer and YUV format sensors.
- Display: Dual display controllers with MIPI-DSI output for LCD panels up to 2540x1600 and HDMI output for external display devices up to 4096x2160. Multiple line pixel storage allows more memory efficient scaling operations and pixel fetching. Hardware display surface rotation is also provided for bandwidth reduction.

In addition to these major elements, Tegra 4 series processors have a broad range of peripheral interfaces to enable communication with wireless baseband, other communications peripherals, audio codecs, power management, and mass storage. When combined with baseband and PMIC chips, the Tegra 4 series processor provides the functionality needed to build a range of low-power devices. Dedicated high-performance mass storage controllers, with their own DMA engines, free the CPU Complex from routine data management tasks.

1.1 The Role of the Technical Reference Manual

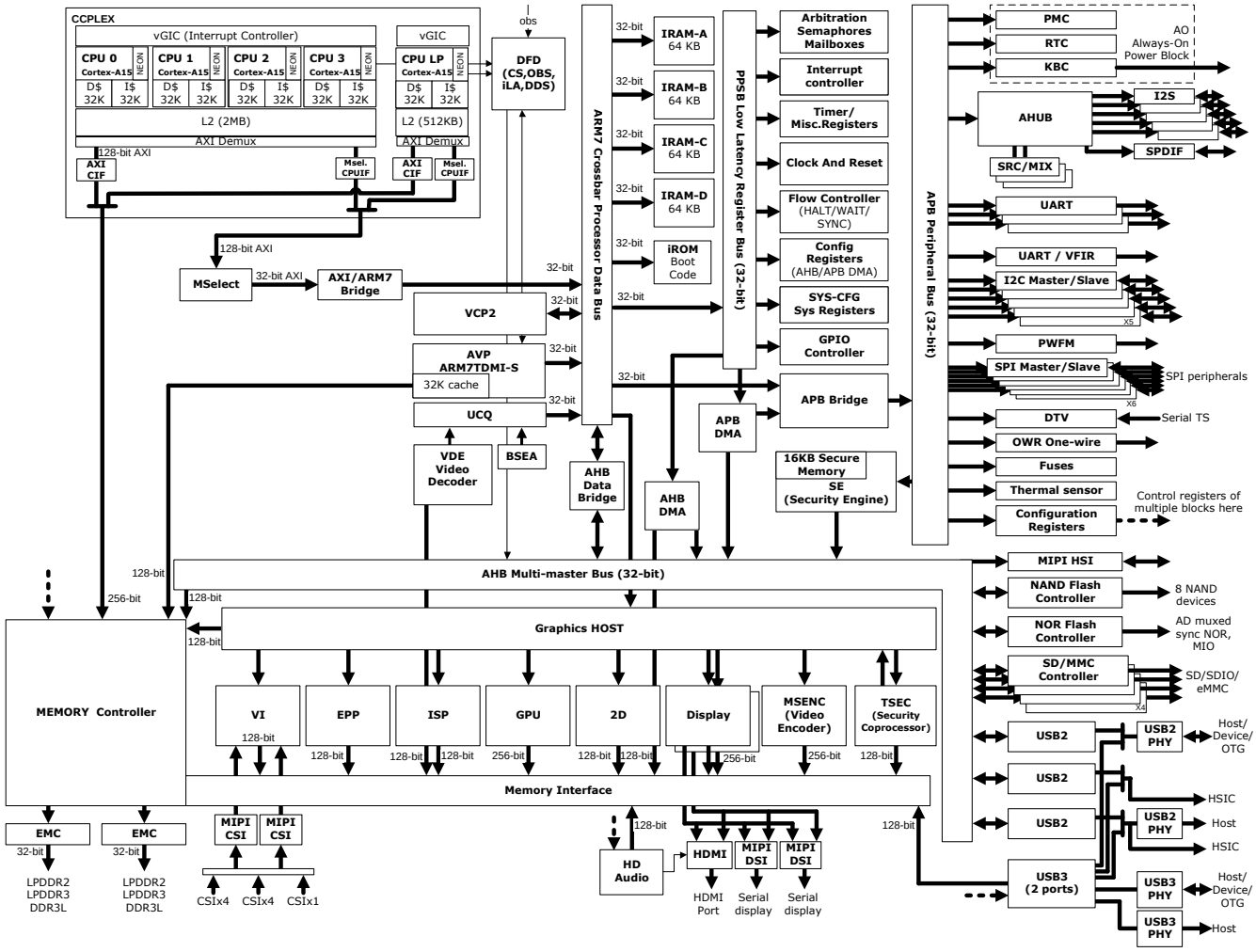
This document is intended to provide guidance to programmers writing code for Tegra 4 devices. It describes the register interfaces and hardware functionality, with the goal to aid anyone in understanding and potentially modifying the NVIDIA provided code.

Some hardware functions described here are not supported by NVIDIA; thus the description of a capability in this document does not necessarily imply software support for that function, nor does it imply that the function has been fully validated by NVIDIA.

1.2 Block Diagram

This diagram provides an overview of a Tegra 4 series processor.

Figure 1: Tegra 4 Series Processor Block Diagram



1.3 Memory Controller and Internal Bus Architecture

The Tegra 4 series processor has a highly optimized dual-channel memory controller, supporting low latency access for the CPU, optimized high bandwidth access for the graphics and video devices, and controlled latency for real time devices such as display.

There is a three-level hierarchy of memory clients:

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

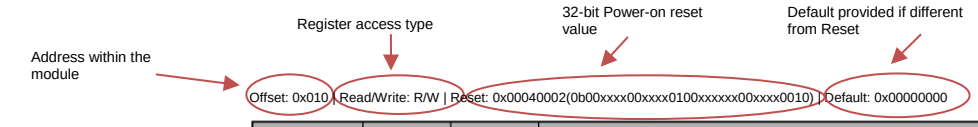
Special provisions are made for the CPU to bypass parts of the memory controller arbitration to help achieve a lower latency.

1.4 Reading Register Tables

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

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

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



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

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

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

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

1.5 Glossary

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

Term	Definition
ADX	Audio Demultiplexer Block, part of the Audio Hub used to demultiplex multiple audio streams.
AHB	AMBA® High-Speed Bus, a multi-master high-speed (relative to APB) bus supporting arbitration and split transactions, defined as part of AMBA 2.
AHBSLVMEM	The AHB slave used for the main memory interface. Acts as an AHB slave and provides a path from there to main memory. Refer to the AHB section.
AMBA	Advanced Microcontroller Bus Architecture, a set of standard buses defined by ARM.
AMX	Audio Multiplexer Block, part of the Audio Hub used to multiplex multiple audio streams together.
APB	AMBA Peripheral Bus, a simple 32-bit single master bus for peripheral devices.
APBDMA	A multi-channel DMA controller for devices on the APB bus, performing DMA between APB and AHB.
ARM	Advanced RISC Machines, a company that licenses CPU IP to Tegra. Also: Architecture Reference Manual, so the ARM ARM defines the CPU architecture.
AVP	Audio-Video Processor, the term used both to describe the ARM7 processor in Tegra devices, and to describe the broader audio and video decode acceleration hardware and RAM associated with the ARM7. Note that the AVP is sometimes known as COP in legacy documentation and registers.
AXI	AMBA Advanced eXtensible Interface, a more advanced bus than AHB defined as part of AMBA 3.
BSEA	Bit Stream Engine for Audio applications
CAR	Clock and Reset module allows controlling clocks and resets to all the modules and subsystems in the Tegra processor.
CEC	Consumer Electronics Control, a part of the HDMI interface specification used for sending device control commands, often from a remote control.
COP	CO-Processor, an obsolete name for the AVP still present in legacy documentation and registers.
CSI	MIPI Camera Serial Interface, a standard high-speed serial interface for connecting cameras to the Tegra processor.
DAM	Digital Audio sample-rate conversion and Mixing block, a block within the Audio Hub that performs audio mixing and sample rate conversion
DSI	MIPI Display Serial Interface, a standard high-speed serial interface for connecting displays to the Tegra processor.
DTV	Digital TV input block, used to stream a serial TV transport stream of compressed data into Tegra, using an SPI like protocol.
DVC	Dynamic Voltage Controller module
EMC	External Memory Controller, a module that interfaces with external DDR/LPDDR devices.
EPP	The video Encoder Pre-Processor, on the Tegra processor capable of filtering, color-space conversion, rotation and pixel format conversion.
GART	Graphic Address Relocation Table, a now obsolete mechanism for mapping from virtual to physical addresses for devices. Remains in Tegra only as an aperture to memory that may be mapped though the SMMU, the replacement for the GART.
GPIO	General Purpose Input/Output, an I/O signal uncommitted to a specific role and controlled by software.
GR2D	2D graphics engine, capable of scaling, filtering and rotating pixel blocks.
HDMI	High-Definition Multimedia Interface, a digital connection carrying uncompressed video and audio at high speed over a single connector.

Term	Definition
HSI	MIPI High-Speed Synchronous Interface, a standard high-speed serial interface for bi-directional communications with baseband processors and other devices.
IDE	Integrated Drive Electronics (or Integrated Device Electronics)
ISP	Image Signal Processor, a hardware engine that is part of the camera processing pipeline.
KBC	Keyboard Controller module allows the Tegra processor to be connected to keyboard matrices of sizes up to 11x8.
MC	Memory Controller module handles requests from internal clients and arbitrates among them to allocate memory bandwidth.
MCCIF or MC-CIF	Memory Controller Client InterFace, the standard interface block between the memory controller sub-system fabric and the client device. Note that some modules may have multiple client interfaces.
MIPI	The Mobile Industry Processor Interface and industry alliance promoting a number of standard interfaces for mobile devices.
MPCore	Multi-processor CPU core, a generic term for a CPU capable of operating as part of an SMP group.
MPE	An older name for the Video Encoder in the Tegra processor capable of encoding raw video stream into MPEG. Now referred to as MSENC.
MSENC	Multi-Standard video Encoder engine.
NAND	A type of flash memory supporting high densities, and commonly used for non-volatile mass storage in portable devices. Accessed in sequential blocks to be generally treated as a file system.
NOR	A type of flash memory, with a direct bus interface that allows random access so code can be executed in place. Generally more costly and less dense than NAND flash memory.
OGL	Open Graphics Library, also known as OpenGL. An API supported on Tegra devices and accelerated in hardware by dedicated 3D and 2D engines.
PCIE	Peripheral Component Interconnect Express, a high-speed interface for external devices connected to the Tegra SOC.
PMC	Power Management Controller module controls the various power management features in the system.
PPSB	PortalPlayer System Bus, a proprietary register bus used for some blocks. Similar to APB. PortalPlayer is a company that was acquired by NVIDIA, and from where parts of Tegra are derived including this bus.
PWFM	Pulse Width Frequency Modulation module generates programmed pulse widths typically used to control backlight in display panels.
PVT	Process, Voltage & Temperature
RISC	Reduced Instruction Set Computer, the CPU architecture used by ARM CPUs.
SDMMC	SD and MMC controller. An I/O controller supporting
SLINK	Serial Link, a legacy and now obsolete name for the SPI controller
SMMU	System Memory Management Unit, a block within the memory controller used to map from a virtual address space to physical addresses for device DMA.
SMP	Symmetric Multi-Processing
SOC	System On a Chip, an integrated circuit containing a CPU, memory controller and the peripheral devices needed for a computing system.
S/PDIF	Sony/Philips Digital Interconnect Format
SPI	Serial Peripheral Interface Bus, a synchronous serial data link, that operates in full duplex mode.
TSEC	Tegra Security co-processor, an embedded security processor used mainly to manage the HDCP encryption and keys on the HDMI link.
TZ	Trust Zone, a secure operating environment of the ARM CPU and the related secure parts of the SOC backbone and devices
TZRAM	Trust Zone secured RAM on the SOC.
UCQ	Unified Command Queue – a sub module within Video Decoder Engine

Term	Definition
VCP2	Vector Co-Processor version 2, a hardware acceleration block for the signal processing parts of audio decode and filtering. Use to offload the ARM7 AVP during audio playback.
VDE	Video Decode Engine, a Tegra hardware acceleration block dedicated to decoding compressed video in various formats.
VI	Video Input block, the acronym used to describe the Tegra block used for camera and related input functions.

[THIS PAGE INTENTIONALLY LEFT BLANK]

2.0 ADDRESS AND INTERRUPT MAP

2.1 System Address Map

Table 1: System Address Map

Description	Address Start	Address End	Default Length
IROM_LOVEC	**0000:0000**	**00ff:ffff**	**16 MB**
PCIE (this address range is used only for DRAM in Tegra® 4 processors, see below for details)	**0100:0000**	**3fff:ffff**	**1008 MB**
PCIE_A1	0100:0000	01ff:ffff	16 MB
PCIE_A2	0200:0000	0fff:ffff	224 MB
PCIE_A3	1000:0000	3fff:ffff	768 MB
Data Memory	**4000:0000**	**40ff:ffff**	**16 MB**
iRAM-A	4000:0000	4000:ffff	64 KB
iRAM-B	4001:0000	4001:ffff	64 KB
iRAM-C	4002:0000	4002:ffff	64 KB
iRAM-D	4003:0000	4003:ffff	64 KB
IRAM_RSVD	**4100:0000**	**47ff:ffff**	**112 MB**
NOR Flash	**4800:0000**	**4fff:ffff**	**128 MB**
NOR Flash_A1	4800:0000	48ff:ffff	16 MB
NOR Flash_A2	4900:0000	4aff:ffff	32 MB
NOR Flash_A3	4b00:0000	4fff:ffff	80 MB
Graphics Host Registers	**5000:0000**	**5002:7fff**	**160 KB**
Host1x	5000:0000	5002:7fff	160 KB
ARM Registers (@ PERIPHBASE)	**5004:0000**	**5005:ffff**	**128 KB**
ARM PERIPHBASE	5004:0000	5005:ffff	128 KB
ARM Interrupt Distributor	5004:1000	5004:1fff	4 KB
AVP CACHE	5004:0000	5004:1fff	8 KB
Interrupt Controller Physical CPU interface	5004:2000	5004:3fff	8 KB
Interrupt Controller Virtual CPU interface (Hypervisor view for requesting CPU)	5004:4000	5004:4fff	4 KB
Interrupt Controller Virtual CPU interface (Hypervisor view for all CPUs)	5004:5000	5004:5fff	4 KB
Interrupt Controller Virtual CPU interface (Virtual Machine view)	5004:6000	5004:7fff	8 KB
MSelect	**5006:0000**	**5006:0fff**	**4 KB**
MSelect Register Space	5006:0000	5006:0fff	4 KB
Verif Aperture (this address range is used for NVIDIA's internal testing, and is simply reserved in production systems)	**5300:0000**	**53ff:ffff**	**16 MB**
Fake Sensor	5300:0000	5300:0fff	4 KB
MPCORE BFM Registers	5300:2000	5300:5fff	16 KB
COP Verification	5300:6000	5300:6fff	4 KB
KBC BFM	5303:2000	5303:2fff	4 KB
I2SBFM0	5303:3000	5303:30ff	256 B
I2SBFM1	5303:3100	5303:31ff	256 B

Description	Address Start	Address End	Default Length
I2SBFM2	5303:3200	5303:32ff	256 B
I2SBFM3	5303:3300	5303:33ff	256 B
I2SBFM4	5303:3400	5303:34ff	256 B
Verif Aperture_RSVD	**5100:0000**	**53ff:ffff**	**48 MB**
Graphics Host	**5400:0000**	**54ff:ffff**	**16 MB**
VI	5408:0000	540b:ffff	256 KB
CSI	5408:0000	540b:ffff	256 KB
EPP	540c:0000	540f:ffff	256 KB
ISP	5410:0000	5413:ffff	256 KB
2D	5414:0000	5417:ffff	256 KB
3D	5418:0000	541b:ffff	256 KB
VG	541c:0000	541f:ffff	256 KB
Display A	5420:0000	5423:ffff	256 KB
Display B	5424:0000	5427:ffff	256 KB
HDMI	5428:0000	542b:ffff	256 KB
TVO	542c:0000	542f:ffff	256 KB
DSI	5430:0000	5433:ffff	256 KB
VS	5434:0000	5437:ffff	256 KB
VCI	5438:0000	543b:ffff	256 KB
DSIB	5440:0000	5443:ffff	256 KB
MSENC	544c:0000	544f:ffff	256 KB
TSEC	5450:0000	5453:ffff	256 KB
Graphics Host_RSVD	**5500:0000**	**56ff:ffff**	**32 MB**
GART	**5700:0000**	**5fff:ffff**	**144 MB**
PPSB	**6000:0000**	**60ff:ffff**	**16 MB**
uP-TAG	6000:0000	6000:0fff	4 KB
Resource Semaphore	6000:1000	6000:1fff	4 KB
Arbitration Semaphore	6000:2000	6000:2fff	4 KB
ARB-PRI	6000:3000	6000:3fff	4 KB
Primary ICTLR	6000:4000	6000:403f	64 B
Primary ICTLR ARB-GNT	6000:4040	6000:40ff	192 B
Secondary ICTLR	6000:4100	6000:413f	64 B
Secondary ICTLR DMA TX	6000:4140	6000:4147	8 B
Secondary ICTLR DMA RX	6000:4148	6000:414f	8 B
Tertiary ICTLR	6000:4200	6000:423f	64 B
Quad ICTLR	6000:4300	6000:433f	64 B
Penta ICTLR	6000:4400	6000:443f	64 B
HIER GROUP1 ICTLR	6000:4800	6000:483f	64 B
TMR	6000:5000	6000:53ff	1 KB
Clock and Reset	6000:6000	6000:6fff	4 KB
Flow Controller	6000:7000	6000:7fff	4 KB
AHB-DMA	6000:8000	6000:9fff	8 KB
AHB-DMA CH0	6000:9000	6000:901f	32 B
AHB-DMA CH1	6000:9020	6000:903f	32 B
AHB-DMA CH2	6000:9040	6000:905f	32 B
AHB-DMA CH3	6000:9060	6000:907f	32 B

Description	Address Start	Address End	Default Length
APB-DMA	6000:a000	6000:bfff	8 KB
APB-DMA CH0	6000:b000	6000:b01f	32 B
APB-DMA CH1	6000:b020	6000:b03f	32 B
APB-DMA CH2	6000:b040	6000:b05f	32 B
APB-DMA CH3	6000:b060	6000:b07f	32 B
APB-DMA CH4	6000:b080	6000:b09f	32 B
APB-DMA CH5	6000:b0a0	6000:b0bf	32 B
APB-DMA CH6	6000:b0c0	6000:b0df	32 B
APB-DMA CH7	6000:b0e0	6000:b0ff	32 B
APB-DMA CH8	6000:b100	6000:b11f	32 B
APB-DMA CH9	6000:b120	6000:b13f	32 B
APB-DMA CH10	6000:b140	6000:b15f	32 B
APB-DMA CH11	6000:b160	6000:b17f	32 B
APB-DMA CH12	6000:b180	6000:b19f	32 B
APB-DMA CH13	6000:b1a0	6000:b1bf	32 B
APB-DMA CH14	6000:b1c0	6000:b1df	32 B
APB-DMA CH15	6000:b1e0	6000:b1ff	32 B
APB-DMA CH16	6000:b200	6000:b21f	32 B
APB-DMA CH17	6000:b220	6000:b23f	32 B
APB-DMA CH18	6000:b240	6000:b25f	32 B
APB-DMA CH19	6000:b260	6000:b27f	32 B
APB-DMA CH20	6000:b280	6000:b29f	32 B
APB-DMA CH21	6000:b2a0	6000:b2bf	32 B
APB-DMA CH22	6000:b2c0	6000:b2df	32 B
APB-DMA CH23	6000:b2e0	6000:b2ff	32 B
APB-DMA CH24	6000:b300	6000:b31f	32 B
APB-DMA CH25	6000:b320	6000:b33f	32 B
APB-DMA CH26	6000:b340	6000:b35f	32 B
APB-DMA CH27	6000:b360	6000:b37f	32 B
APB-DMA CH28	6000:b380	6000:b39f	32 B
APB-DMA CH29	6000:b3a0	6000:b3bf	32 B
APB-DMA CH30	6000:b3c0	6000:b3df	32 B
APB-DMA CH31	6000:b3e0	6000:b3ff	32 B
System Registers	6000:c000	6000:c2ff	768 B
AHB Arbitration + Gizmo Controller			336 B
AHB/APB Debug Bus			176 B
Secure Boot			256 B
STAT-MON	6000:c400	6000:c7ff	1 KB
Activity Monitor	6000:c800	6000:cbff	1 KB
GPIO-1	6000:d000	6000:d0ff	256 B
GPIO-2	6000:d100	6000:d1ff	256 B
GPIO-3	6000:d200	6000:d2ff	256 B
GPIO-4	6000:d300	6000:d3ff	256 B
GPIO-5	6000:d400	6000:d4ff	256 B
GPIO-6	6000:d500	6000:d5ff	256 B
GPIO-7	6000:d600	6000:d6ff	256 B

Description	Address Start	Address End	Default Length
GPIO-8	6000:d700	6000:d7ff	256 B
VCP	6000:e000	6000:efff	4 KB
Exception vectors	6000:f000	6000:ffff	4 KB
AVPUCQ	6001:0000	6001:00ff	256 B
BSEA	6001:1000	6001:1fff	4 KB
VDE	6001:a000	6001:dbff	15 KB
SXE	6001:a000	6001:afff	4 KB
BSEV	6001:b000	6001:bfff	4 KB
MBE	6001:c000	6001:c0ff	256 B
PPE	6001:c200	6001:c2ff	256 B
MCE	6001:c400	6001:c4ff	256 B
TFE	6001:c600	6001:c6ff	256 B
PPB	6001:c800	6001:c8ff	256 B
VDMA	6001:ca00	6001:caff	256 B
UCQ	6001:cc00	6001:ccff	256 B
FRAMEID	6001:d800	6001:dbff	1 KB
IPATCH	6001:dc00	6001:dfff	1 KB
PPSB_RSVD	**6100:0000**	**67ff:ffff**	**112 MB**
External I/O	**6800:0000**	**68ff:ffff**	**16 MB**
MIO/EXIO	6800:0000	68ff:ffff	16 MB
MIOBFM_SPDIF			256 B
MIOBFM_I2C			256 B
MIOBFM_DIVISOR			256 B
MIOBFM_UART			256 B
MIOBFM_VFIR			256 B
MIOBFM_DVCI			256 B
MIOBFM_XMB			256 B
MIOBFM_SLINK			256 B
MIOBFM_UCQ			256 B
MIOBFM_PWM			256 B
MIOBFM_PUPD_GPIO			256 B
MIOBFM_MPM_GPIO			128 B
MIOBFM_MPM_DPD			128 B
MIOBFM_TSENSOR			256 B
MIOBFM_USB			256 B
MIOBFM_RSVD0			256 B
MIOBFM_OWL			256 B
MIOBFM_HDA			256 B
MIOBFM_I2S0			256 B
MIOBFM_I2S1			256 B
MIOBFM_SATA			256 B
MIOBFM_PMC			256 B
MIOBFM_CEC			256 B
MIOBFM_I2S2			256 B
MIOBFM_I2S3			256 B
MIOBFM_I2S4			256 B

Description	Address Start	Address End	Default Length
MIOBFM_GPIO			1 KB
MIOBFM_CAR			4 KB
MIOBFM_AUDIO			256 B
External IO_RSVD	**6900:0000**	**6fff:ffff**	**112 MB**
APB	**7000:0000**	**70ff:ffff**	**16 MB**
MISC	7000:0000	7000:3fff	16 KB
PP			1 KB
SC1X_PADS			1 KB
GP			1 KB
SECURE_REGS			256 B
SATA_AUX			256 B
PINMUX_AUX			4 KB
UART-A	7000:6000	7000:603f	64 B
UART-B	7000:6040	7000:607f	64 B
VFIR	7000:6100	7000:61ff	256 B
UART-C	7000:6200	7000:62ff	256 B
UART-D	7000:6300	7000:63ff	256 B
UART-E (currently unassigned)	7000:6400	7000:64ff	256 B
PCIE_X2_0_IOBIST	7000:6800	7000:68ff	256 B
PCIE_X2_1_IOBIST	7000:6900	7000:69ff	256 B
PCIE_X4_IOBIST	7000:6a00	7000:6aff	256 B
SATA_IOBIST	7000:6c00	7000:6dff	512 B
Verification APB Memory (Sim only)	7000:7000	7000:7fff	4 KB
NAND Controller	7000:8000	7000:81ff	512 B
HSMMC/CE-ATA	7000:8500	7000:85ff	256 B
XIO Interface	7000:8a00	7000:8bff	512 B
Sync NOR	7000:9000	7000:9fff	4 KB
PWM Controller	7000:a000	7000:a0ff	256 B
MIPIHSI Baseband	7000:b000	7000:bfff	4 KB
I2C	7000:c000	7000:c0ff	256 B
TWC	7000:c100	7000:c1ff	256 B
DTV	7000:c300	7000:c3ff	256 B
I2C2	7000:c400	7000:c4ff	256 B
I2C3	7000:c500	7000:c5ff	256 B
OWR	7000:c600	7000:c6ff	256 B
I2C4	7000:c700	7000:c7ff	256 B
I2C5	7000:d000	7000:d0ff	256 B
SPI-1	7000:d400	7000:d5ff	512 B
SPI-2	7000:d600	7000:d7ff	512 B
SPI-3	7000:d800	7000:d9ff	512 B
SPI-4	7000:da00	7000:dbff	512 B
SPI-5	7000:dc00	7000:ddff	512 B
SPI-6	7000:de00	7000:dfff	512 B
RTC	7000:e000	7000:e0ff	256 B
KBC	7000:e200	7000:e2ff	256 B
PMC	7000:e400	7000:ebff	2 KB

Description	Address Start	Address End	Default Length
MC	7001:9000	7001:9fff	4 KB
EMC	7001:b000	7001:b7ff	2 KB
FUSE	7000:f800	7000:fbff	1 KB
KFUSE	7000:fc00	7000:ffff	1 KB
LA	7001:0000	7001:1fff	8 KB
SE	7001:2000	7001:3fff	8 KB
TSENSOR	7001:4000	7001:4fff	4 KB
CEC	7001:5000	7001:5fff	4 KB
ATOMICS	7001:6000	7001:7fff	8 KB
MC0	7001:8000	7001:8fff	4 KB
MC1	7001:c000	7001:cfff	4 KB
MCB	7001:9000	7001:9fff	4 KB
EMC0	7001:a000	7001:a7ff	2 KB
EMC1	7001:a800	7001:afff	2 KB
EMCB	7001:b000	7001:b7ff	2 KB
SATA	7002:0000	7002:ffff	64 KB
XUSB_PADCTL	7009:f000	7009:ffff	4 KB
XUSB_HOST	7009:0000	7009:9fff	40 KB
XUSB_DEV	700d:0000	700d:9fff	40 KB
DDS	700a:0000	700a:11ff	4608 B
HDA	7003:0000	7003:ffff	64 KB
CSITE	7004:0000	7007:ffff	256 KB
AUDIO_CLUSTER	7008:0000	7008:1fff	8 KB
APBIF			512 B
AUDIO			256 B
I2S0			256 B
I2S1			256 B
I2S2			256 B
I2S3			256 B
I2S4			256 B
DAM0			256 B
DAM1			256 B
DAM2			256 B
SPDIF			256 B
AMX0			256 B
AMX1			256 B
ADX0			256 B
ADX1			256 B
APBIF2			512 B
SPEEDO	700c:0000	700c:7fff	32 KB
SPEEDO_0			256 B
SPEEDO_1			256 B
SPEEDO_PMON	700c:8000	700c:ffff	32 KB
SPEEDO_PMON_0			512 B
SPEEDO_PMON_1			512 B
DP2	700e:0000	700e:00ff	256 B

Description	Address Start	Address End	Default Length
APB2JTAG	700e:1000	700e:11ff	512 B
SOC_THERM	700e:2000	700e:2fff	4 KB
MIPI_CAL	700e:3000	700e:30ff	256 B
SYSCTR0	700f:0000	700f:ffff	64 KB
SYSCTR1	7010:0000	7010:ffff	64 KB
DVFS	7011:0000	7011:03ff	1 KB
APB_RSVD	**7100:0000**	**77ff:ffff**	**112 MB**
AHB_A1	**7800:0000**	**78ff:ffff**	**16 MB**
SDMMC-1	7800:0000	7800:01ff	512 B
SDMMC-1B	7800:1000	7800:11ff	512 B
SDMMC-2	7800:0200	7800:03ff	512 B
SDMMC-2B	7800:2200	7800:23ff	512 B
SDMMC-3	7800:0400	7800:05ff	512 B
SDMMC-3B	7800:3400	7800:35ff	512 B
SDMMC-4	7800:0600	7800:07ff	512 B
SDMMC-4B	7800:4600	7800:47ff	512 B
AHB_A1_RSVD	**7900:0000**	**7bff:ffff**	**48 MB**
AHB_A2	**7c00:0000**	**7dff:ffff**	**32 MB**
PPCS (AHB to MC flush)	7c00:0000	7c00:ffff	64 KB
TZRAM	7c01:0000	7c01:ffff	64 KB
USB	7d00:0000	7d00:17ff	6 KB
USB2	7d00:4000	7d00:57ff	6 KB
USB3	7d00:8000	7d00:97ff	6 KB
AHB_A2_RSVD	**7e00:0000**	**7fff:ffff**	**32 MB**
External Memory	**8000:0000**	**ffef:ffff**	**2047 MB**
EMEM	8000:0000	ffef:ffff	2047 MB
IROM (boot code)	**fff0:0000**	**fff0:ffff**	**64 KB**
Hi-VEC	**ffff:0000**	**ffff:ffff**	**64 KB**
IROM_HIVEC	**fff0:0000**	**ffff:ffff**	**1 MB**
Non-Aperture Interrupts			
SW_INTR			
MPCORE			
External PMIC			

2.2 Available DRAM Address Ranges

Table 2 describes the various address regions that are available as DRAM in Tegra 4 devices.

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

Clients/Controllers are logically (static) grouped in different SWNAME/SWID. SMMU translation can be enabled/disabled for individual SWNAME. Table 3 gives the Client to SW Name mapping details.

Note: MMIO in Table 2 below refers to the MMIO target, register, or non-DRAM region. Not all of the MMIO regions actually have an MMIO target.
Physical/Virtual Address Range is available for Host1X clients for all apertures.

Table 2: DRAM Address Ranges

Aperture	Range	Size (MB)	H/W Config Option	Recommended Config Value	Physical DRAM Range			IOVA Range[1]
					2 GB	3GB	4 GB	
T35_IROM,HiVEC	FFF0_FFFF - FFFF_FFFF	1	Selectable	MMIO	No	No	No	No
DRAM	8000_0000 - FFEF_FFFF	2047	Always DRAM	N/A	Yes	Yes	Yes	Yes
AHB_A2_rsvd	7E00_0000 - 7FFF_FFFF	32	Selectable	DRAM	No	No	Yes	Yes
AHB_A2	7C00_0000 - 7DFF_FFFF	32	Always MMIO	N/A	No	No	No	No
AHB_A1_rsvd	7900_0000 - 7BFF_FFFF	48	Selectable	DRAM	No	No	Yes	Yes
AHB_A1	7800_0000 - 78FF_FFFF	16	Selectable	MMIO	No	No	No	No
APB_rsvd	7100_0000 - 77FF_FFFF	112	Selectable	DRAM	No	No	Yes	Yes
APB	7000_0000 - 70FF_FFFF	16	Always MMIO	N/A	No	No	No	No
ExtIO_rsvd	6900_0000 - 6FFF_FFFF	112	Selectable	DRAM	No	No	Yes	Yes
ExtIO	6800_0000 - 68FF_FFFF	16	Always MMIO	N/A	No	No	No	No
PPSB_rsvd	6100_0000 - 67FF_FFFF	112	Selectable	DRAM	No	No	Yes	Yes
PPSB	6000_0000 - 60FF_FFFF	16	Always MMIO	N/A	No	No	No	No
GART/GPU_GART	5700_0000 - 5FFF_FFFF	144	Selectable	DRAM	No	No	Yes	Yes
Graphics Host rsvd (HOST1X_DR_RSVD)	5500_0000 - 56FF_FFFF	32	Selectable	DRAM	No	No	Yes	Yes
Graphics Host (HOST1X_DR)	5400_0000 - 54FF_FFFF	16	Always MMIO	N/A	No	No	No	No
Verif Aper + Rsvd	5100_0000 - 53FF_FFFF	48	Selectable	DRAM	No	No	Yes	Yes
Host1x+PERIPH	5000_0000 - 50FF_FFFF	16	Always MMIO	N/A	No	No	No	No
NOR_A3	4B00_0000 - 4FFF_FFFF	80	Selectable	DRAM	No	No	Yes	Yes
NOR_A2	4900_0000 - 4AFF_FFFF	32	Selectable	DRAM	No	No	Yes	Yes
NOR_A1	4800_0000 - 48FF_FFFF	16	Selectable	DRAM	No	No	Yes	Yes
IRAM_rsvd	4100_0000 - 47FF_FFFF	112	Selectable	DRAM	No	No	Yes	Yes
IRAM	4000_0000 - 40FF_FFFF	16	Always MMIO	N/A	No	No	No	No
PCIE_A3	1000_0000 - 3FFF_FFFF	768	Selectable	DRAM	No	Yes	Yes	Yes
PCIE_A2	0200_0000 - 0FFF_FFFF	224	Selectable	DRAM	No	Yes	Yes	Yes
PCIE_A1	0100_0000 - 01FF_FFFF	16	Selectable	DRAM	No	Yes	Yes	Yes
T40_IROM,LoVEC	0000_0000 - 00FF_FFFF	16	Selectable	MMIO	No	No	No	No

Notes:

[1] The IOVA Range column is available for AVP or AHB clients that have SMMU translation enabled or PA address for AHB clients that have SMMU translation disabled.

Note: Requests from all clients (except main CPU) are allowed to use SMMU translation. Any request address range that can reach the MC can be translated by the SMMU.
SMMU is a sub-unit in the memory controller. Address ranges marked as "No" in table would not reach the memory controller.
All Host1x clients can see the full 4GB physical or IOVA range

Table 3: Client to SW Name Mapping

SW Group	Client	Description
AVPC	AVP Cache	ARM7 Audio-Video Processor (AVP)
DC	Display head 0	Display reads, head 0
DCB	Display head 1	Display reads, head 1
VDE	VDE	Video Decode Engine
MSENC	MSENC	Multi Standard Video Encoder
G2	GR2D	2D engine
HC	Host1x	Host interface
HDA	HDA	High-definition Audio engine
VI	VI	Video Input engine for CSI
ISP	ISPW	ISP
EPP	EPP	EPP Engine
MPCORE	AXICIF	Eagle CPU cores
MPCORELP	AXICIF_LP	Eagle Shadow CPU core
NV	3D engine	3D Engine
PPCS	AHBDMA, AHBSLV	Clients in the AHB cluster (ppcs2mc_swid=1'b0)
PPCS1	AHBDMA, AHBSLV	Same as PPCS, ppcs2mc_swid=1'b0)
PTC	MC (SMMU)	Misses from SMMU PTC
XUSB_HOST	USB3 Host	Although the same driver is expected to control both units, each has its own SWNAME
XUSB_DEV	USB3 device	
TSEC	TSEC	Tegra Security co-processor

Note: Requests from all clients (except main CPU) are allowed to use SMMU translation. Any request address range that can reach the MC can be translated by the SMMU.

The clients behind AHBSLV in the above table are:

- SDMMC
- USB1/2/3
- BSEA
- DDS
- SE
- BSEV
- NOR
- NAND
- AHBDMA
- ARC
- CORESIGHT
- VCP
- COP
- CPU

- APBDMA clients, which are also behind AHBSLVMEM (APBDMA_Ch{0-31})

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

2.3 Interrupt Mapping

The next five tables show the mapping of the interrupts from system devices to the bit fields in the interrupt controllers.

Interrupts to the CPU's embedded interrupt controller (GIC) are in this same order, but start at offset 32 as the first 32 are reserved for the CPU's internal interrupts. See the Interrupt Controller section of this document for more information on how interrupts are routed.

Table 4: Primary Interrupt Controller Mapping

Global Interrupt Number	Interrupt Target	Legacy Interrupt Controller (LIC) Name	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description	Interrupt Order
31		PRI_ICTLR	31	SDMMC4	SDMMC	SDMMC4 Controller	0
30		PRI_ICTLR	30	OWR	OWR	OWR Interrupt	0
29	CPU	PRI_ICTLR	29	ARB_SEM_GNT_CPU	Semaphore	GNT.0 Arbitration Grant Status of CPU	0
28	COP	PRI_ICTLR	28	ARB_SEM_GNT_COP	Semaphore	GNT.1 Arbitration Grant Status of COP	0
27	CPU	PRI_ICTLR	27	AHB_DMA_CPU	APB_DMA	AHB DMA Controller (CPU)	0
26	CPU	PRI_ICTLR	26	APB_DMA_CPU	APB_DMA	APB Bridge DMA Controller (CPU)	0
25		PRI_ICTLR	25	VCP	VCP	VCP	0
24		PRI_ICTLR	24	NANDCTRL	NANDCTRL	NAND Flash Controller	0
23		PRI_ICTLR	23	SATA_CTL	SATA	SATA Controller Interrupt	0
22		PRI_ICTLR	22	Currently Unmapped		Currently Unassigned	0
21		PRI_ICTLR	21	USB2	USB	USB Device	0
20		PRI_ICTLR	20	USB	USB	USB Device	0
19		PRI_ICTLR	19	SDMMC3	SDMMC	SDMMC3 Controller	0
18		PRI_ICTLR	18	AVP_UCQ	AVP_UCQ	AVP UCQ Interrupt	0
17		PRI_ICTLR	17	VDE	VDE	VDE Interrupt	0
16		PRI_ICTLR	16	Currently Unmapped		Currently Unassigned	0
15		PRI_ICTLR	15	SDMMC2	SDMMC	SDMMC2 Controller	0
14		PRI_ICTLR	14	SDMMC1	SDMMC	SDMMC1 Controller	0
13		PRI_ICTLR	13	SATA_RX_STAT	SATA	SATA RX Wake-Up Interrupt	1
12		PRI_ICTLR	12	VDE_SXE	VDE	VDE SXE Interrupt	3
11		PRI_ICTLR	11	VDE_BSEA	VDE	AVP BSEA Interrupt	0
10		PRI_ICTLR	10	VDE_BSEV	VDE	VDE BSE-V Interrupt	1
9		PRI_ICTLR	9	VDE_SYNC_TOKEN	VDE	VDE Sync Token Interrupt	0
8		PRI_ICTLR	8	VDE_UCQ	VDE	VDE UCQ Error Interrupt	4
7	CPU	PRI_ICTLR	7	SHR_SEM_OUTBOX_ EMPTY	Semaphore	CPU Out Box Empty Interrupt	3
6	COP	PRI_ICTLR	6	SHR_SEM_OUTBOX_FULL	Semaphore	COP Out Box Full Interrupt	2
5	COP	PRI_ICTLR	5	SHR_SEM_INBOX_EMPTY	Semaphore	COP In Box Empty Interrupt	1
4	CPU	PRI_ICTLR	4	SHR_SEM_INBOX_FULL	Semaphore	CPU In Box Full Interrupt	0
3		PRI_ICTLR	3	CEC	CEC	CEC General Interrupt	0
2		PRI_ICTLR	2	RTC	RTC	RTC Interrupt	0
1		PRI_ICTLR	1	TMR2	Timer	TMR2 Interrupt	0
0		PRI_ICTLR	0	TMR1	Timer	TMR1 Interrupt	0

Table 5: Secondary Interrupt Controller Mapping

Global Interrupt Number	Interrupt Target	Legacy Interrupt Controller (LIC) Name	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description	Interrupt Order
63		SEC_ICTLR	31	Currently Unmapped		Currently Unassigned	0
62		SEC_ICTLR	30	CLDVFS	CL-DVFS	Close Loop DVFS control logic	0
61	COP	SEC_ICTLR	29	AHB_DMA_COP	AHB_DMA	AHB-DMA Interrupt (COP)	1
60	COP	SEC_ICTLR	28	APB_DMA_COP	APB_DMA	APB-DMA Interrupt (COP)	1
59		SEC_ICTLR	27	SBC1	SPI	SPI Controller (SBC1)	0
58		SEC_ICTLR	26	SE	SE	SE and TZRAM General Interrupt	0
57		SEC_ICTLR	25	USB3_DEV_PME	USB3	USB3_DEV_PME	0
56		SEC_ICTLR	24	USB3_DEV_SMI	USB3	USB3_DEV_SMI	0
55		SEC_ICTLR	23	GPIO5	GPIO (external)	GPIO4 Interrupt	0
54		SEC_ICTLR	22	STAT_MON	sts_stat_mon	SYS_STATS_MON	0
53		SEC_ICTLR	21	I2C5	I2C	I2C5 Interrupt	0
52		SEC_ICTLR	20	VFIR	VFIR	VFIR Controller	0
51		SEC_ICTLR	19	EDP	SOC_THERM	EDP (Electrical Design Power) interrupt	0
50		SEC_ICTLR	18	TSEC	TSEC	Tegra HDCP Security Controller	0
49		SEC_ICTLR	17	XUSB_PADCTL	SUSB	XUSB (USB3) pad interrupt	0
48		SEC_ICTLR	16	THERMAL	SOC_THERM	Thermal interrupt	0
47		SEC_ICTLR	15	HSI	HIS	MIPI HSI baseband controller	0
46		SEC_ICTLR	14	UARTC	UART	UART-C	0
45		SEC_ICTLR	13	ACTMON	ActMon	ACTMON	0
44		SEC_ICTLR	12	USB3_DEV_HOST	USB3	USB3_DEV_HOST	1
43		SEC_ICTLR	11	USB3_HOST_PME	USB3	USB3_HOST_PME	0
42		SEC_ICTLR	10	TMR4	Timer	TMR4 Interrupt	0
41		SEC_ICTLR	9	TMR3	Timer	TMR3 Interrupt	0
40		SEC_ICTLR	8	USB3_HOST_SMI	USB3	USB3_HOST_SMI	0
39		SEC_ICTLR	7	USB3_HOST_INT	USB3	USB3_HOST_INT	0
38		SEC_ICTLR	6	I2C	I2C	I2C Interrupt	0
37		SEC_ICTLR	5	UARTB	UART	UART-B	0
36		SEC_ICTLR	4	UARTA	UART	UART-A	0
35		SEC_ICTLR	3	GPIO4	GPIO (external)	GPIO3 Interrupt	0
34		SEC_ICTLR	2	GPIO3	GPIO (external)	GPIO2 Interrupt	0
33		SEC_ICTLR	1	GPIO2	GPIO (external)	GPIO1 Interrupt	0
32		SEC_ICTLR	0	GPIO1	GPIO (external)	GPIO0 Interrupt	0

Table 6: Tertiary Interrupt Controller Mapping

Global Interrupt Number	Interrupt Target	Legacy Interrupt Controller (LIC) Name	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description	Interrupt Order
95		TRI_ICTLR	31	SW_INTR	N/A (Software)	SW Controlled Interrupt	0
94		TRI_ICTLR	30	SBC5	SPI	SPI Controller (SBC5)	0
93		TRI_ICTLR	29	SBC4	SPI	SPI Controller (SBC4)	0

Global Interrupt Number	Interrupt Target	Legacy Interrupt Controller (LIC) Name	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description	Interrupt Order
92		TRI_ICTLR	28	I2C3	I2C	I2C3 Interrupt	0
91		TRI_ICTLR	27	Former UARTE	UART	Former UART-E (currently unassigned)	0
90		TRI_ICTLR	26	UARTD	UART	UART-D	0
89		TRI_ICTLR	25	GPIO7	GPIO (external)	GPIO7 Interrupt	0
88		TRI_ICTLR	24	Currently Unmapped		Currently Unassigned	0
87		TRI_ICTLR	23	GPIO6	GPIO (external)	GPIO6 Interrupt	0
86		TRI_ICTLR	22	PMU_EXT	PMIC	External Power Management Chip Interrupt	0
85		TRI_ICTLR	21	KBC	KBC	KBC Interrupt	0
84		TRI_ICTLR	20	I2C2	I2C	I2C2 Interrupt	0
83		TRI_ICTLR	19	SBC3	SPI	SPI Controller (SBC3)	0
82		TRI_ICTLR	18	SBC2	SPI	SPI Controller (SBC2)	0
81		TRI_ICTLR	17	HDA	HDA	HDA Interrupt	0
80		TRI_ICTLR	16	Currently Unmapped		Currently Unassigned	0
79		TRI_ICTLR	15	SBC6	SPI	SPI Controller (SBC6)	0
78		TRI_ICTLR	14	EMC	EMC	EMC General Interrupt	0
77		TRI_ICTLR	13	MC	MC	MC General Interrupt	0
76		TRI_ICTLR	12	Currently Unmapped		Currently Unassigned	0
75		TRI_ICTLR	11	HDMI	HDMI	HDMI INT from pins	0
74		TRI_ICTLR	10	DISPLAYB	DISPLAYB	Display B General Interrupt	0
73		TRI_ICTLR	9	DISPLAY	DISPLAY	Display A General Interrupt	0
72		TRI_ICTLR	8	GR2D	GR2D	2D General Interrupt	0
71		TRI_ICTLR	7	ISP	ISP	ISP General Interrupt	0
70		TRI_ICTLR	6	EPP	EPP	EPP General Interrupt	0
69		TRI_ICTLR	5	VI	VI	VI General Interrupt	0
68		TRI_ICTLR	4	MPE	MPE	MPE General Interrupt	0
67	CPU	TRI_ICTLR	3	HOST1X_GEN_CPU	HOST1X	MPCORE General Interrupt	3
66	COP1	TRI_ICTLR	2	HOST1X_GEN_COP	HOST1X	COP General Interrupt	2
65	CPU	TRI_ICTLR	1	HOST1X_SYNCPT_CPU	HOST1X	MPCORE SyncPt Interrupt	1
64	COP1	TRI_ICTLR	0	HOST1X_SYNCPT_COP	HOST1X	COP SyncPt Interrupt	0

Table 7: Quaternary Interrupt Controller Mapping

Global Interrupt Number	Interrupt Target	Legacy Interrupt Controller (LIC) Name	LIC Interrupt Number	Interrupt Name	Source Block	Description	Interrupt Order
127		QUAD_ICTLR	31	HIER_GROUP1_ CPU	HIER_GROUP1	CPU interrupt from hierarchical group1.	0
126		QUAD_ICTLR	30	CAR	CAR	CAR Interrupt	0
125		QUAD_ICTLR	29	GPIO8	GPIO (external)	GPIO8 Interrupt	0
124		QUAD_ICTLR	28	WDT_AVP	Timer	Watchdog AVP Interrupt	2
123		QUAD_ICTLR	27	WDT_CPU	Timer	Watchdog CPU Interrupt	1
122		QUAD_ICTLR	26	HIER_GROUP1_ COP	HIER_GROUP1	COP interrupt from hierarchical group1.	0
121		QUAD_ICTLR	25	TMR5	Timer	TMR5 Interrupt	0
120		QUAD_ICTLR	24	I2C4	I2C	I2C4 Interrupt	0
119		QUAD_ICTLR	23	APB_DMA_CH15	APB_DMA	APB-DMA Ch 15	0
118		QUAD_ICTLR	22	APB_DMA_CH14	APB_DMA	APB-DMA Ch 14	0
117		QUAD_ICTLR	21	APB_DMA_CH13	APB_DMA	APB-DMA Ch 13	0
116		QUAD_ICTLR	20	APB_DMA_CH12	APB_DMA	APB-DMA Ch 12	0
115		QUAD_ICTLR	19	APB_DMA_CH11	APB_DMA	APB-DMA Ch 11	0
114		QUAD_ICTLR	18	APB_DMA_CH10	APB_DMA	APB-DMA Ch 10	0
113		QUAD_ICTLR	17	APB_DMA_CH9	APB_DMA	APB-DMA Ch 9	0
112		QUAD_ICTLR	16	APB_DMA_CH8	APB_DMA	APB-DMA Ch 8	0
111		QUAD_ICTLR	15	APB_DMA_CH7	APB_DMA	APB-DMA Ch 7	0
110		QUAD_ICTLR	14	APB_DMA_CH6	APB_DMA	APB-DMA Ch 6	0
109		QUAD_ICTLR	13	APB_DMA_CH5	APB_DMA	APB-DMA Ch 5	0
108		QUAD_ICTLR	12	APB_DMA_CH4	APB_DMA	APB-DMA Ch 4	0
107		QUAD_ICTLR	11	APB_DMA_CH3	APB_DMA	APB-DMA Ch 3	0
106		QUAD_ICTLR	10	APB_DMA_CH2	APB_DMA	APB-DMA Ch 2	0
105		QUAD_ICTLR	9	APB_DMA_CH1	APB_DMA	APB-DMA Ch 1	0
104		QUAD_ICTLR	8	APB_DMA_CH0	APB_DMA	APB-DMA Ch 0	0
103		QUAD_ICTLR	7	AUDIO_CLUSTER	AUDIO_ CLUSTER	Audio Cluster Interrupt	0
102		QUAD_ICTLR	6	TSENSOR	TSENSOR	Tsensor interrupt	0
101		QUAD_ICTLR	5	AVP_CACHE	ARM7 Cache	ARM7 Cache Interrupt	0
100		QUAD_ICTLR	4	Former PCIE_WAKE	PCIE	PCIE wake up signal (currently unassigned)	0
99		QUAD_ICTLR	3	Former PCIE_MSI	PCIE	PCIE message-signaled interrupt (currently unassigned)	0
98		QUAD_ICTLR	2	Former PCIE_INT	PCIE	PCIE catch-all error/legacy interrupts (currently unassigned)	0
97		QUAD_ICTLR	1	USB3	USB	USB Device	0
96		QUAD_ICTLR	0	SNOR	SNOR	Sync NOR Controller	0

Table 8: Fifth Interrupt Controller Mapping

Global Interrupt Number	Interrupt Target	Legacy Interrupt Controller (LIC) Name	LIC Interrupt Number	Interrupt Name	Source Block	Interrupt Description	Interrupt Order
159		PENTA_ICTLR	31	Currently Unmapped		Currently Unassigned	0
158		PENTA_ICTLR	30	Currently Unmapped		Currently Unassigned	0
157		PENTA_ICTLR	29	Currently Unmapped		Currently Unassigned	0
156		PENTA_ICTLR	28	TMR0	Timer	TMR0 Interrupt	0
155		PENTA_ICTLR	27	TMR9	Timer	TMR9 Interrupt	0
154		PENTA_ICTLR	26	TMR8	Timer	TMR8 Interrupt	0
153		PENTA_ICTLR	25	TMR7	Timer	TMR7 Interrupt	0
152		PENTA_ICTLR	24	TMR6	Timer	TMR6 Interrupt	0
151		PENTA_ICTLR	23	SDMMC4_SYS	SDMMC	SDMMC4 int. for standard driver (e.g., WOA)	0
150		PENTA_ICTLR	22	SDMMC3_SYS	SDMMC	SDMMC3 int. for standard driver (e.g., WOA)	0
149		PENTA_ICTLR	21	SDMMC2_SYS	SDMMC	SDMMC2 int. for standard driver (e.g., WOA)	0
148		PENTA_ICTLR	20	SDMMC1_SYS	SDMMC	SDMMC1 int. for standard driver (e.g., WOA)	0
147		PENTA_ICTLR	19	CPU3_PMU_INTR	A15 CPU3	CPU3 PMIC Interrupt	0
146		PENTA_ICTLR	18	CPU2_PMU_INTR	A15 CPU2	CPU2 PMIC Interrupt	0
145		PENTA_ICTLR	17	CPU1_PMU_INTR	A15 CPU1	CPU1 PMIC Interrupt	0
144		PENTA_ICTLR	16	CPU0_PMU_INTR	A15 CPU0	CPU0 PMIC Interrupt	0
143		PENTA_ICTLR	15	APB_DMA_CH31	APB_DMA	APB-DMA Ch 31	0
142		PENTA_ICTLR	14	APB_DMA_CH30	APB_DMA	APB-DMA Ch 30	0
141		PENTA_ICTLR	13	APB_DMA_CH29	APB_DMA	APB-DMA Ch 29	0
140		PENTA_ICTLR	12	APB_DMA_CH28	APB_DMA	APB-DMA Ch 28	0
139		PENTA_ICTLR	11	APB_DMA_CH27	APB_DMA	APB-DMA Ch 27	0
138		PENTA_ICTLR	10	APB_DMA_CH26	APB_DMA	APB-DMA Ch 26	0
137		PENTA_ICTLR	9	APB_DMA_CH25	APB_DMA	APB-DMA Ch 25	0
136		PENTA_ICTLR	8	APB_DMA_CH24	APB_DMA	APB-DMA Ch 24	0
135		PENTA_ICTLR	7	APB_DMA_CH23	APB_DMA	APB-DMA Ch 23	0
134		PENTA_ICTLR	6	APB_DMA_CH22	APB_DMA	APB-DMA Ch 22	0
133		PENTA_ICTLR	5	APB_DMA_CH21	APB_DMA	APB-DMA Ch 21	0
132		PENTA_ICTLR	4	APB_DMA_CH20	APB_DMA	APB-DMA Ch 20	0
131		PENTA_ICTLR	3	APB_DMA_CH19	APB_DMA	APB-DMA Ch 19	0
130		PENTA_ICTLR	2	APB_DMA_CH18	APB_DMA	APB-DMA Ch 18	0
129		PENTA_ICTLR	1	APB_DMA_CH17	APB_DMA	APB-DMA Ch 17	0
128		PENTA_ICTLR	0	APB_DMA_CH16	APB_DMA	APB-DMA Ch 16	0

2.4 Hierarchical Groups

The next table is a hierarchical interrupt grouping which generates interrupts for the first level of interrupt controllers.

Table 9: Hierarchical Interrupt Grouping

Hierarchical Interrupt Controller Name	Interrupt Number	Interrupt Name	Source Block	Interrupt Description
HIER_GROUP1	31 - 15	Currently Unmapped		Currently Unassigned
HIER_GROUP1	14	MPCORE_CTIIRQ3		CPU3 CTI interrupt
HIER_GROUP1	13	MPCORE_CTIIRQ2		CPU2 CTI interrupt
HIER_GROUP1	12	MPCORE_CTIIRQ1		CPU1 CTI interrupt
HIER_GROUP1	11	MPCORE_CTIIRQ0		CPU0 CTI interrupt
HIER_GROUP1	10	TMR_SHARED	Timer	TMR_SHARED interrupt
HIER_GROUP1	9	FLOW_RSM_COP	FlowCtlr	FLOW - RSM1 Resume for COP or COP1
HIER_GROUP1	8	FLOW_RSM_CPU	FlowCtlr	FLOW - RSM0 Resume for CPU
HIER_GROUP1	7	Former SPEEDO_PMON_1	SPEEDOS	Speedo Interrupt (currently unassigned)
HIER_GROUP1	6	Former SPEEDO_PMON_0	SPEEDOS	Speedo Interrupt (currently unassigned)
HIER_GROUP1	5	EVENT_GPIO_D	GPIO (External)	Event Detector GPIO Port D
HIER_GROUP1	4	EVENT_GPIO_C	GPIO (External)	Event Detector GPIO Port C
HIER_GROUP1	3	EVENT_GPIO_B	GPIO (External)	Event Detector GPIO Port B
HIER_GROUP1	2	EVENT_GPIO_A	GPIO (External)	Event Detector GPIO Port A
HIER_GROUP1	1	MPCORE_INTERRIRQ	CCPLEX	CPU MPCore INTERRIRQ
HIER_GROUP1	0	MPCORE_AXIERRIRQ	CCPLEX	CPU MPCore AXIERRIRQ

[THIS PAGE INTENTIONALLY LEFT BLANK]

3.0 INTERRUPT CONTROLLER

This section describes the architecture for routing and handling of the Tegra® 4 family processor interrupts. It also describes semaphores, which provide a mechanism for the processors to arbitrate for the use of various resources.

From the perspective of interrupts: devices (GPU, Memory Controller, Video encode/decode engines, and various I/O devices) are the sources of interrupts; and processors are the target of interrupts. From sources, interrupts go to interrupt controllers which, based on configuration, prioritize and route them to appropriate target processor. There are two different types of interrupt controllers used in Tegra, the ARM® vGIC in the main CPU complex and the Legacy Interrupt Controller (LIC), which is used for the AVP and also to control wake events for the CPU.

Glossary and Acronyms

Acronym	Description
GIC	Generic Interrupt Controller (also known as GICv1, part of Cortex™-A15 MPCore™)
vGIC	Virtual GIC (also known as GICv2, part of Cortex-A15 MPCore)
LIC	Legacy Interrupt Controller (used for the ARM7 AVP)
WFI	Wait For Interrupt (an ARM instruction)
WFE	Wait For Event (an ARM instruction)
IRQ	Interrupt Request
FIQ	Fast Interrupt Request
IPI	Inter Processor Interrupt
SGI	Software Generated Interrupt
PPI	Private Peripheral Interrupts
SPI	Shared Peripheral Interrupts
SMP	Symmetric MultiProcessors
MSI	Message Signaled Interrupt
CPU	Unless specified otherwise, it refers to main (Cortex-A15) CPUs
COP	Refers to ARM7 AVP

3.1 Functional Description

3.1.1 Interrupt Handling Mechanism

There are two different types of interrupt controllers (vGIC, LIC) which receive interrupts from devices (i.e. hardware interrupts aka SPIs) or processors (i.e., software interrupts including IPI , also known as SGIs), arbitrate them and send them to appropriate target processor(s). From interrupts perspective, there are two different types of processors: CPUs (Cortex™A15), and COP (ARM7 AVP). The vGIC is the interrupt controller for the Cortex-A15 CPUs, and the LIC is for the ARM7 AVP. Any processor can initiate a software interrupt, targeted to any one or more processors (including self). However, IPIs can only be initiated by a Cortex-A15 CPU to any one or more Cortex-A15 CPUs (including self).

There are a total of 160 hardware interrupts in Tegra 4 devices. Interrupt sources are allocated one or more interrupt as required. The 160 interrupts are grouped into slices of 32, where each slice can be configured independently.

3.1.1.1 ARM Processors' IRQ and FIQ

ARM processors (Cortex-A15 and ARM7) have two input pins to receive two different types of interrupts. These interrupts are called IRQ (Interrupt Request) and FIQ (Fast Interrupt Request). The interrupts are implemented as active-low pins on the ARM processor input, thus processor pins are named nIRQ and nFIQ.

The ARM processor goes into the IRQ mode or the FIQ mode depending up on which interrupt is activated. Conventionally, FIQ is used as secure interrupt (although either IRQ or FIQ can be configured as secure interrupt). As a result, non-secure interrupts are normally IRQ only. FIQ is relatively low latency interrupt, so interrupts which require low latency, or are time critical, can potentially be configured as FIQ.

3.1.1.2 Interrupt Controllers

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

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

The LIC (Legacy Interrupt Controller) is primarily used for COP (ARM7). But it is also used for generating interrupt as wake events for CPUs when vGIC is powered off. All of the device hardware interrupt signals are sent to the LIC first, which routes them to COP as well as forwards them to other interrupt controllers. The LIC also provides a software set/clear mechanism for all of the interrupts.

The interrupt controller used for CPUs (Cortex-A15) is called vGIC (virtual generic interrupt controller). The vGIC is SMP interrupt controller. It receives interrupts targeted to any one or more of the CPUs in the SMP complex. Similar to Tegra 3 which had one GIC per CPU cluster, there is one vGIC per CPU cluster.

3.1.1.3 vGIC Interrupt Sources

The vGIC can support up to 256 interrupts. All interrupts are identified by a unique ID. There are three types of interrupt sources vGIC.

Software Generated Interrupts (SGIs)

These are software interrupts which are generated by writing to GIC (or vGIC) register. A maximum of 16 SGIs, with ID0-15, can be generated by software. These are also referred to as IPI (Inter Processor Interrupts).

Private Peripheral Interrupts (PPIs)

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

There were five possible PPIs for each Cortex-A9 processor: global timer (ID27), legacy nFIQ (ID28), private timer interrupt (ID29), watchdog timer interrupt (ID30), and legacy nIRQ (ID31).

There are seven PPIs for each Cortex-A15 processor: virtual maintenance interrupt (ID25), hypervisor timer interrupt (ID26), virtual timer interrupt (ID27), legacy nFIQ (ID28), secure physical timer interrupt (ID29), non-secure physical timer interrupt (ID30), legacy nIRQ (ID31).

Tegra 4 devices do not use legacy nFIQ/nIRQ of vGIC or GIC.

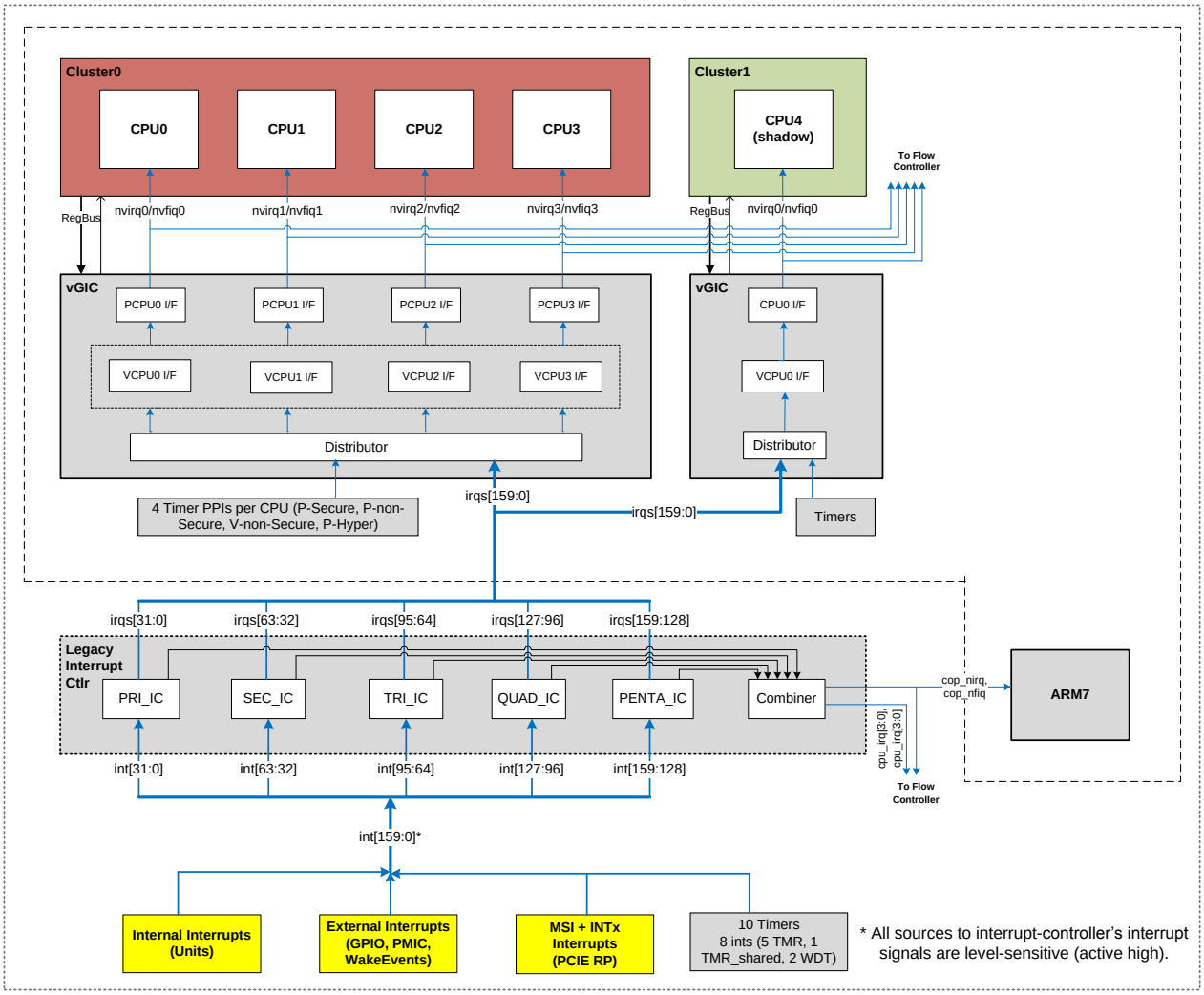
Shared Peripheral Interrupts (SPIs)

SPIs are external hardware interrupts which are generated by asserting signals on vGIC input pins (called IRQS). From Tegra hardware perspective, these are the interrupts which are generated by various units to be sent to interrupt controllers. There can be a maximum of 224 SPIs generated through IRQS[223:0] pins. However, Tegra 4 devices use only 160.

3.1.1.4 Interrupt Routing

Tegra hardware interrupts (also known as Shared Peripheral Interrupts) can be generated either by Internal SoC units, or external events (through GPIO, USB. etc.). All of these hardware interrupts are routed through sideband active-high level-sensitive signals to LIC. Devices which natively generate MSIs are converted by their bus wrappers to sideband interrupt signals.

Figure 2: Interrupt Routing (Top-Level View)



As shown in the figure above, hardware interrupts are routed to the interrupt controllers (called IC32) within LIC, where each IC32 handles 32 interrupts. The interrupt signals are synchronized within LIC to the system clock. The synchronized signals are combined with software set/clear bits per interrupt. This allows software to set or clear individual interrupts, provided the corresponding interrupt is not asserted by hardware. The resulting interrupt signals are broadcast to LIC, for further processing, and vGIC. Within LIC, and vGIC. per interrupt and per CPU enable-masks are configured to qualify the interrupts targeted for the corresponding processor. This mechanism allows any interrupt to be targeted to any one or more processors in Tegra 4 devices.

For the details of interrupt handling within LIC and vGIC, refer to the following sections.

3.1.1.5 Interrupt Handling with Targeted CPU and vGIC Powered Off

When a CPU and associated vGIC are powered off (e.g., CPU rail is powered off for cluster0), then the legacy-interrupt-controller is configured to act as a "proxy" interrupt controller to generate IRQ/FIQ for that CPU. Also, Tegra timers (instead of vGIC timers) will have to be set up if a timer interrupt is required. The flow-controller uses legacy interrupt controller IRQ/FIQ to power-up targeted CPU (and vGIC).

Following sequence describes the steps to follow to be able to handle interrupts when CPU and vGIC are powered off:

- [SW-CPU] Prepare CPU (including vGIC) for power off

    - Flush caches and save context
    - Setup Tegra timer (if required)
    - Configure corresponding legacy-interrupt controller
    - Configure flow-controller for interrupt (from flow-controller) wake-up event
    - Trigger power-off by WFI/WFE instruction
- [HW-Flow] Power off CPU with the help of PMC
- [HW-Flow] Wait for wake-up interrupt (from legacy-interrupt-controller)
- [HW-Flow] At interrupt assertion, power up CPU with the help of PMC
- [SW-CPU] Restore the context (including vGIC context)
- [SW-CPU] CPU receives the interrupt and jumps to interrupt handler

Since Tegra interrupts are level-triggered, the interrupt that triggered the wake-up is still asserted and is now visible to vGIC. vGIC will process the interrupt and send it to CPU.

3.1.1.6 Interrupt Handling with Targeted CPU Powered Off

Each CPU can be individually powered off while corresponding vGIC is powered on. In this case, the flow controller uses vGIC interrupts to wake up the CPU. Note, this allows vGIC timers, and IPI interrupts to be used as wake events.

Following sequence describes the steps to follow to be able to handle interrupts when targeted CPU is powered off:

- [SW-CPU] Prepare CPU for power off
    - Flush caches and save context
    - Configure flow-controller for interrupt (from vGIC) wake-up event
    - Trigger power-off by WFI/WFE instruction
- [HW-Flow] Power off CPU with the help of PMC
- [HW-Flow] Wait for wake-up interrupt (from vGIC)
- [HW-Flow] At interrupt assertion, power up CPU with the help of PMC
- [SW-CPU] Restore the context
- [SW-CPU] CPU receives the interrupt and jumps to interrupt handler

Note that since the vGIC IRQ/FIQ are level-triggered, the interrupt which triggered the wake-up is still asserted.

3.1.2 Virtual Generic Interrupt Controller (vGIC)

This section provides brief description of Cortex-A15 vGIC (also known as GICv2). Also, it assumes that you are aware of GIC also known as GICv1 (see previous section for Cortex-A9 GIC description). For more details of vGIC, refer to Cortex-A15 TRM (for vGIC implementation specific spec) and/or vGIC architectural spec.

Similar to GIC, vGIC consists of the physical CPU interface and the physical distributor. In addition, to support CPU virtualization it consists of an entirely new component (the virtual CPU Interface) to be configured by the Hypervisor and used by the VM (virtual machine).

By default, the vGIC is configured to behave as GIC. In systems without a hypervisor, vGIC functionality is not expected to be used beyond GIC.

3.1.2.1 Functional Description

IRQ and FIQ Bypass Disable

Prior to powering off a CPU (e.g., in LP2 entry), its IRQ/FIQ needs to be disabled at vGIC CPU-interface. At the same time, vGIC has to generate IRQ/FIQ equivalent interrupts for CPU wake purpose. In GICv1, IRQ/FIQ interrupt lines going to CPU are same lines which are also used for wake-interrupt. Therefore, we cannot disable the vGIC CPU interface if we need to use same IRQ/FIQ for CPU wake purposes. This effectively means that IRQ/FIQ to CPU cannot be disabled, which creates a race condition for CPU power off entry. Also, in GICv1, disabling GICv1 CPU-interface puts GICv1 into bypass mode such that incoming (from SoC to GICv1) legacy IRQ/FIQ interrupts are sent to CPU when GICv1 is disabled (also known as bypassed), which means disabling GIC CPU-interface doesn't really disable IRQ/FIQ to CPU.

To fix above race issue, vGIC (GICv2) provides a wake IRQ/FIQ (for wake-event purpose), which are not disabled by GIC CPU-interface disable. Also, vGIC supports disabling of the CPU from it respective IRQ/FIQ interrupt lines, while still providing GIC-generated IRQ and FIQ functionality as wakeup events to SoC (system power controller). This is done by adding a bypass override capability when the interrupts are disabled at the CPU Interface. To support this, four register control bits are added to the Physical CPU Interface Control Register (also known as GICC_CTLR in GICv2 and ICDDCR in GICv1). Since the VCPUIF is not involved in power-management, these bits do not exist in the VCPU interface.

Table 10: CPU Interface Control Register (GICC_CTLR) – Secure View

Bit Field Name	Bit	Description
EOImodeNS	10	Unrelated to bypass disable (used when interrupts are virtualized).
EOImode	9	Unrelated to bypass disable (used when interrupts are virtualized).
IRQBypDisNS	8	When the CPU Interface is disabled, prevents the legacy bypass IRQ signal being forwarded to the CPU (if IRQs are under Non-secure control).
FIQBypDisNS	7	When the CPU Interface is disabled, prevents the legacy bypass FIQ signal being forwarded to the CPU (if FIQs are under Non-secure control).
IRQBypDisable	6	When the CPU Interface is disabled, prevents the legacy bypass IRQ signal being forwarded to the CPU (if IRQs are under Secure control).
FIQBypDisable	5	When the CPU Interface is disabled, prevents the legacy bypass FIQ signal being forwarded to the CPU (if FIQs are under Secure control).
Various	4:2	Reserved. Same as GICv1.
Enable Non-secure	1	Same as GICv1 0 = disables all Non-secure interrupts control bits in the distributor from changing state because of any external stimulus change that occurs on the corresponding **SPI** or **PPI** signals 1 = enables the distributor to update register locations for Non-secure interrupts
Enable secure	0	Same as GICv1 0 = disables all Secure interrupt control bits in the distributor from changing state because of any external stimulus change that occurs on the corresponding **SPI** or **PPI** signals. 1 = enables the distributor to update register locations for Secure interrupts.

In GICv1 (i.e., in Tegra 3), GIC CPU Interface Control Register is programmed to 0x0 (secure view) to disable the GIC CPU interface. In GICv2, GIC CPU Interface Control Register needs to be programmed to 0x1E0 (secure view) to disable GIC CPU-interface. Depending upon FIQ/IRQ being secure or non-secure, this programming can be different.

Note: Non-secure banked version of above register does not have bits<8:7>. For more details, refer to "IRQ and FIQ bypass disable functionality" in the vGIC specification.

Split EOI Functionality

Within GICv1, every interrupt has a priority, and any interrupt being active will prevent another interrupt of equal (or lower) priority being asserted to the CPU. This causes problems for virtualization where there is not necessarily any well-defined priority for interrupts between the various VMs. To address this problem the functionality associated with EOI is divided into two distinct functions – "Priority Drop" to be used after the interrupt is acknowledged to drop the running priority of the CPU

back down but leave the interrupt in the ACTIVE state, and “Deactivate Interrupt” to be used when the interrupt handling is truly completed by the VM, causing the interrupt to transition from ACTIVE to INACTIVE in the physical Distributor. To avoid Hypervisor intervention being necessary when a Guest OS completes interrupt processing, the virtual CPU Interface is able to directly trigger a Deactivate Interrupt event on the interrupt distributor.

Virtual CPU Interface

The Virtual CPU Interface is a new block of hardware (vs. GIC) which is added to support CPU virtualization. Its only connection to the existing GIC is via the interrupt deactivation interface described above section.

The Virtual CPU Interface provides two sets of registers for each CPU in the system. The “front end” registers are intended to be mapped into the VM’s IPA space where the Guest OS’ CPU Interface registers are expected to reside. The “back end” registers are intended to be accessed by the Hypervisor only. The front and back end registers are mapped into separate 4KB address space. The “back end” registers primarily comprise a list of active and pending interrupts for the current Virtual CPU. These registers are updated by the Hypervisor when new interrupts occur, and by the VCPUIF itself in response to accesses to the “front end” from the VM.

The “front end” registers have exactly the same format as the existing GIC CPU Interface, but reflect the status of interrupts stored in the list registers rather than physical interrupts reported by the distributor.

Security and Virtual FIQ

In ARM virtualization, Hypervisor and all VMs run in TrustZone™ Non-Secure domains. Therefore, all vGIC features are available to Non-Secure accesses. Protection of the physical GIC components and the Hypervisor view of the Virtual CPU Interface is achieved using MMU protection. However, some operating systems running as Guests might require the security features of the GIC architecture to allow interrupts to be divided into IRQ and FIQ. To support this, the Hypervisor and Virtual CPU Interface provide a view compatible with the Secure view of the physical GIC, even though the software is running in the Non-Secure domain. This means that the vGIC interface includes various “security” related bits which have no relationship to TrustZone but exist purely to support this model.

Life of an Interrupt in vGIC

Physical Interrupt Asserted

The physical interrupt is routed to the vGIC, which generates an interrupt which is taken by the Hypervisor. The Hypervisor reads INTACK on the physical vGIC to determine the interrupt number. Determining that the interrupt is destined for a virtual machine, the EOI register is written to prevent further interrupts being masked by the vGIC.

Hypervisor locates an empty list register (in the target VM’s virtual CPU interface) and writes a new value indicating that this is a Valid interrupt in PENDING state, with the physical and virtual interrupt numbers and its priority as assigned by the VM.

Virtual Interrupt Asserted

Upon the write to the list register, the VCPU interface will re-evaluate whether to assert the Virtual IRQ signal, based on the priority of the interrupt, any currently active interrupts and the current setting of the Priority Mask register. Assuming these conditions are met the Virtual Interrupt signal will be asserted. When the CPU returns to the virtual machine, (with the CPSR.I bit clear), the Virtual Interrupt will be taken triggering a jump to the normal IRQ vector.

Virtual Interrupt Acknowledged

The Guest OS’s interrupt handler will access the VCPU Interface’s INTACK register. In response to this, the VCPU interface determines the highest priority PENDING interrupt stored in the list. Its priority is checked against the Active Priorities register and the Priority Mask register. Assuming that the pending interrupt’s priority is higher, its virtual interrupt number is returned, its status in the list register changed to ACTIVE and the corresponding bit in the Active Priorities register is set. Otherwise, the spurious interrupt value is returned. Once the Guest OS has received the virtual interrupt number it is able to execute interrupt processing exactly as it would if it were running on bare hardware with a real GIC.

Virtual Interrupt EOI

Once interrupt processing is complete, the Guest OS writes to the EOI register on the VCPU interface to indicate this. In response to this, the VCPU interface determines the highest priority ACTIVE interrupt from the list registers, and the highest priority set bit in the Active Priorities register. This highest priority set bit is cleared. Assuming a highest priority ACTIVE interrupt was found, a Deactivate Interrupt signal is sent to the Interrupt Distributor specifying the physical interrupt number from the list register. The list register is updated to indicate that the entry is now empty (INVALID).

Timers

The Cortex-A15 processor provides a set of four timers for each processor in the cluster.

- Physical Timer for use in Secure and Non-secure PL1 modes. The registers for the Physical Timer are banked to provide Secure and Non-secure copies
- Virtual Timer for use in Non-secure PL1 modes.
- Physical Timer for use in Hypervisor mode.

The counter value is distributed to the processor with a synchronous binary encoded 64-bit bus, **CNTVALUEB[63:0]**. Within each processor, a set of Timer registers are allocated to the CP15 coprocessor space.

3.1.2.2 vGIC Clock Frequency

The vGIC can operate at any integer multiple that is slower than the main Cortex-A15 CPU clock (CLK) using the PERIPHCLKEN signal. For example, the CLK to internal vGIC clock frequency ratio can be 2:1 or 3:1. PERIPHCLKEN asserts one CLK cycle prior to the rising edge of the internal IC clock. The CLK to internal vGIC clock frequency ratio can be changed dynamically using PERIPHCLKEN.

Both CPU clusters (cluster0 and cluster1) incorporate their own vGIC. The vGIC will be running at half the CPU CLK frequency of the cluster it is in.

3.1.2.3 vGIC Context Save/Restore

The vGIC (and Timers) is part of Cortex-A15 MPCore non-CPU (including L2 Cache) logic. The cluster0 vGIC would be powered by CPU rail, and cluster1 is powered by SoC rail. Also, if non-CPU is power gated then vGIC would be power gated as part of non-CPU partition.

In general, vGIC context need to be saved/restored in following cases.

- When CPU(s) is individually power gated (i.e., LP2 power state), then vGIC is still powered, so vGIC context doesn't have to be saved/restored. Therefore, CPUs can be individually powered-off (i.e., put into LP2 state) and powered-on without saving/restoring their vGIC context.
- When non-CPU partition is power-gated or the CPU rail is powered off, then the context of corresponding vGIC needs to be saved.
- When one cluster is migrated to other cluster, then VGIC context need to be migrated from source cluster to destination cluster.
- When SoC rail is powered off (i.e., LP0 power state), then vGIC will be powered off. So vGIC has to be configured/restored at LP0 exit (also known as warm boot)
- When using virtualization extension of vGIC, then for switching from one VM to another Hypervisor will save the contents of all the "back end" registers (i.e., VM view of VCPU Interface registers) from the outgoing VM and load them in the registers for the incoming VM. This restores the VCPU interface to the correct state for the incoming VM.

When a VCPU is migrated (context switched) to another physical CPU, the software has to save the contents of all the "back end" registers (i.e. VM view of VCPU Interface registers) associated with the outgoing CPU, and load them to the corresponding registers of the incoming CPU.

3.1.3 Arbitration Semaphores

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

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

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

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

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

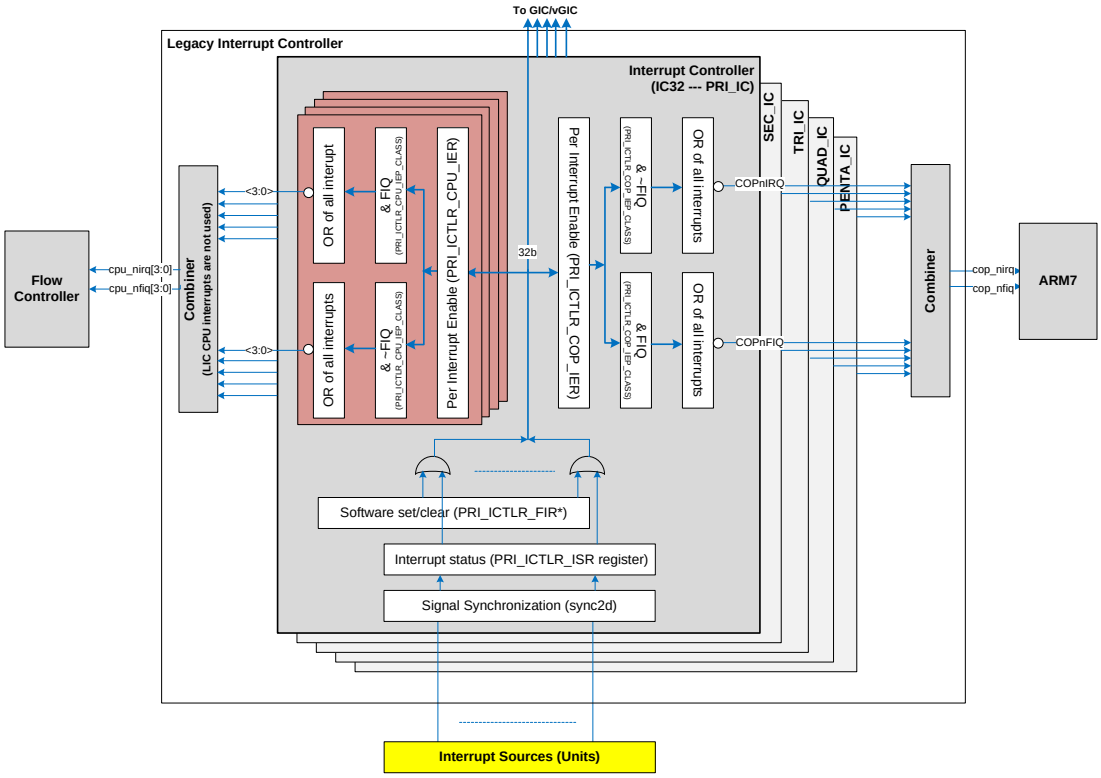
3.1.4 Interrupts Used for Processor Power Ups

The interrupt signals (nIRQ and nFIQ per processor), which are routed to each processor, are also routed to the Flow Controller unit. Based on configuration, the Flow Controller uses them to power up corresponding CPU. Refer to the Flow Controller section in this TRM for the details of processor wake-up.

3.2 Legacy Interrupt Controller (LIC)

Interrupts from various Tegra 4 units are routed to the LIC. The LIC consists of five interrupt controllers (called IC32), where each IC32 handles 32 interrupts (so a total of 160 interrupts). Each IC32 generates five sets of IRQ/FIQ interrupts, one for COP (ARM7), and other one for CPU<3:0>. The interrupts for CPU<3:0> are used by flow controller only for the purpose of wake events.

Figure 3: Legacy Interrupt Controller Block Diagram



The following text discusses interrupts as they relate to the COP IRQ/FIQ, but unless specified otherwise, this also applies to the CPU<3:0> IRQ/FIQ interrupts.

3.2.1.1 LIC Functional Description

Each of IC32 in the LIC handles 32 interrupts. As shown in above figure, the IC32 synchronizes the incoming hardware interrupt requests and combines (bitwise OR) them with software interrupts (FIR). The software interrupt can set be set/clear by FIR register. The combined software/hardware interrupts signal are broadcast to GIC, vGIC, and are used by IC32 for targeting interrupts to COP. This allows any interrupt to be routed to any one or more processors (Cortex-A15/ARM7) in Tegra system. Note, routing can be changed dynamically. In addition, any of these interrupts can be mapped to either IRQ or FIQ.

The interrupt routing (to COP) is achieved by configuring the Interrupt Enable Register (IER) and the Interrupt Class Registers (IEP_CLASS). Each IC32 has a set of IER and IEP Class registers. The COP registers are called COP_IER and COP_IEP_CLASS. Similarly the CPU IER/IEP Class Registers are called CPU_IER and CPU_IEP_CLASS.

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

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

The nIRQ signals generated by five IC32 controllers are combined (logical AND) in the combiner to generate final nIRQ (called nirq1 in above diagram) which is routed to COP. Similarly nFIQ signals are combined (logical AND) in the combiner to generate nFIQ which is routed to COP.

3.2.1.2 LIC Registers Description

The LIC consists of five interrupt controllers (called IC32), where each IC32 handles 32 interrupts (so a total of 160 interrupts). Each IC32 defines two sets of registers: one set for COP (ARM7) and other set for CPU (unused).

Each IC32 has 16 32-bit registers. The COP and CPU<n> interrupt enable registers (COP_IER/CPU_IER) indicate the interrupts enabled for the COP and CPU<3:0> respectively. These registers have their respective SET and CLR registers that allow setting or clearing individual bits of these registers without the need of a Read-Modify-Write cycle.

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

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

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

The difference between ISR and VFIQ/VIRQ registers are:

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

The VIRQ_COP and VIRQ_CPU registers are calculated as follows.

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

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

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

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

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

3.2.1.3 Hierarchical Interrupts Grouping

To reduce the number of interrupts (as seen by interrupt controllers also known as IRQS), interrupt sources which are non-critical or used in pre-production systems only are grouped together. The hierarchical interrupt group (HIER_GROUPx) groups 32 interrupts into two interrupts: one for CPU (called HIER_GROUPx_CPU) and other one for COP (called HIER_GROUPx_COP). There is a software configurable enable register (HIER_GROUPx_{CPU,COP}_ENABLE) for each of the two interrupts which can be independently programmed to select interrupts for that destination. Also, there is a status register (HIER_GROUPx_{CPU,COP}_STATUS) for each of the two interrupts which provides the status of interrupt which are qualified by enable register. Note, incoming interrupt signals are active-high level signals, so interrupts have to cleared (by software) at the source unit. The interrupts are grouped as follows.

```
HIER_GROUP1_CPU_STATUS[31:0] = HIER_GROUP1_CPU_ENABLE[31:0] & source_INT[31:0]
HIER_GROUP1_CPU = | HIER_GROUP1_CPU_STATUS[31:0]     //OR-ing of all bits

HIER_GROUP1_COP_STATUS[31:0] = HIER_GROUP1_COP_ENABLE[31:0] & source_INT[31:0]
HIER_GROUP1_COP = | HIER_GROUP1_COP_STATUS[31:0]    //OR-ing of all bits
```

The Address Interrupt section of this document lists the interrupt sources which are in the hierarchical interrupts group(s).

3.2.2 Primary Interrupt Controller Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

3.2.2.1 PRI_ICTLR_VIRQ_CPU_0

Valid Interrupt Request Status for CPU Register

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

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC

Bit	Reset	Description
1	X	TMR2
0	X	TMR1

3.2.2.2 PRI_ICTLR_VIRQ_COP_0

Valid Interrupt Status for COP Register

Offset: 0x4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC

Bit	Reset	Description
1	X	TMR2
0	X	TMR1

3.2.2.3 PRI_ICTLR_VFIQ_CPU_0

FIQ Valid Interrupt Status for CPU Register

Offset: 0x8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC

Bit	Reset	Description
1	X	TMR2
0	X	TMR1

3.2.2.4 PRI_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for COP Register

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

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC

Bit	Reset	Description
1	X	TMR2
0	X	TMR1

3.2.2.5 PRI_ICTLR_ISR_0

Latched Interrupt Status Register (HW)

Offset: 0x10 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC

Bit	Reset	Description
1	X	TMR2
0	X	TMR1

3.2.2.6 PRI_ICTLR_FIR_0

Forced Interrupt Status Register (SW)

Offset: 0x14 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC

Bit	Reset	Description
1	X	TMR2
0	X	TMR1

3.2.2.7 PRI_ICTLR_FIR_SET_0

Force Interrupt Register Set

Writing a 1 in any bit position will set the corresponding interrupt.

Offset: 0x18 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC

Bit	Reset	Description
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.8 PRI_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

Writing a 1 in any bit position will clear the corresponding Interrupt.

Offset: 0x1c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL

Bit	Reset	Description
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.9 PRI_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

Offset: 0x20 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL

Bit	Reset	Description
3	X	CEC
2	X	RTC
1	X	TMR2
0	X	TMR1

3.2.2.10 PRI_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU.

Offset: 0x24 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL

Bit	Reset	Description
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
31:0	0x0	CPU_IER_SET:
0	0x0	TMR1

3.2.2.11 PRI_ICTLR_CPU_IER_CLR_0

CPUs Clear Interrupt Enable for CPU

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU

Offset: 0x28 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL

Bit	Reset	Description
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.12 PRI_ICTLR_CPU_IEP_CLASS_0

CPUs Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU. 1 = FIQ, 0 = IRQ.

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY

Bit	Reset	Description
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.13 PRI_ICTLR_COP_IER_0

Enabled Interrupt Source for COP Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x30 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ

Bit	Reset	Description
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC
1	X	TMR2
0	X	TMR1

3.2.2.14 PRI_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for COP.

Offset: 0x34 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN

Bit	Reset	Description
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.15 PRI_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

Writing a 1 in any bit position will disable the corresponding Interrupt Source for COP.

Offset: 0x38 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV

Bit	Reset	Description
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.16 PRI_ICTLR_COP_IEP_CLASS_0

COPs Interrupt Enable Priority Class (FIQ/IRQ) Register

Set Priority Interrupt Source for COP. 1 = FIQ, 0 = IRQ.

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV

Bit	Reset	Description
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.17 PRI_ICTLR_VIRQ_CPU1_0

Valid Interrupt Request Status for CPU1 Register

Flags set by hardware, cleared by software.

Offset: 0x60 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA

Bit	Reset	Description
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC
1	X	TMR2
0	X	TMR1

3.2.2.18 PRI_ICTLR_VFIQ_CPU1_0

FIQ Valid Interrupt Status for CPU1 Register

Flags set by hardware, cleared by software.

Offset: 0x64 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE

Bit	Reset	Description
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC
1	X	TMR2
0	X	TMR1

3.2.2.19 PRI_ICTLR_CPU1_IER_0

Enabled Interrupt Source for CPU1 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x68 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE

Bit	Reset	Description
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC
1	X	TMR2
0	X	TMR1

3.2.2.20 PRI_ICTLR_CPU1_IER_SET_0

Set Interrupt Enable for CPU1 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU1.

Offset: 0x6c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1

Bit	Reset	Description
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.21 PRI_ICTLR_CPU1_IER_CLR_0

CPU1s Clear Interrupt Enable for CPU1

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU1.

Offset: 0x70 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2

Bit	Reset	Description
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.22 PRI_ICTLR_CPU1_IEP_CLASS_0

CPU1s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source For CPU1. 1 = FIQ, 0 = IRQ.

Offset: 0x74 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE

Bit	Reset	Description
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.23 PRI_ICTLR_VIRQ_CPU2_0

Valid Interrupt Request Status for CPU2 Register

Flags set by Hardware, cleared by software.

Offset: 0x78 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ

Bit	Reset	Description
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC
1	X	TMR2
0	X	TMR1

3.2.2.24 PRI_ICTLR_VFIQ_CPU2_0

FIQ Valid Interrupt Status for CPU2 Register

Flags set by Hardware, cleared by software

Offset: 0x7c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3

Bit	Reset	Description
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC
1	X	TMR2
0	X	TMR1

3.2.2.25 PRI_ICTLR_CPU2_IER_0

Enabled Interrupt Source for CPU2 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x80 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB

Bit	Reset	Description
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC
1	X	TMR2
0	X	TMR1

3.2.2.26 PRI_ICTLR_CPU2_IER_SET_0

Set Interrupt Enable for CPU2 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU2.

Offset: 0x84 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2

Bit	Reset	Description
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.27 PRI_ICTLR_CPU2_IER_CLR_0

CPU2s Clear Interrupt Enable for CPU2

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU2.

Offset: 0x88 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL

Bit	Reset	Description
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.28 PRI_ICTLR_CPU2_IEP_CLASS_0

CPU2s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU2. 1 = FIQ, 0 = IRQ.

Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP

Bit	Reset	Description
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.29 PRI_ICTLR_VIRQ_CPU3_0

Valid Interrupt Request Status for CPU3 Register

Flags set by Hardware, cleared by software.

Offset: 0x90 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU

Bit	Reset	Description
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC
1	X	TMR2
0	X	TMR1

3.2.2.30 PRI_ICTLR_VFIQ_CPU3_0

FIQ Valid Interrupt Status for CPU3 Register

Flags set by Hardware, cleared by software.

Offset: 0x94 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP
27	X	AHB_DMA_CPU

Bit	Reset	Description
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC
1	X	TMR2
0	X	TMR1

3.2.2.31 PRI_ICTLR_CPU3_IER_0

Enabled Interrupt Source for CPU3 Register

Interrupt Enable Status. 0 = Disabled

Offset: 0x98 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR
29	X	ARB_SEM_GNT_CPU
28	X	ARB_SEM_GNT_COP

Bit	Reset	Description
27	X	AHB_DMA_CPU
26	X	APB_DMA_CPU
25	X	VCP
24	X	NANDCTRL
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
15	X	SDMMC2
14	X	SDMMC1
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV
9	X	VDE_SYNC_TOKEN
8	X	VDE_UCQ
7	X	SHR_SEM_OUTBOX_EMPTY
6	X	SHR_SEM_OUTBOX_FULL
5	X	SHR_SEM_INBOX_EMPTY
4	X	SHR_SEM_INBOX_FULL
3	X	CEC
2	X	RTC
1	X	TMR2
0	X	TMR1

3.2.2.32 PRI_ICTLR_CPU3_IER_SET_0

Set Interrupt Enable for CPU3 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU3.

Offset: 0x9c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU

Bit	Reset	Description
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.33 PRI_ICTLR_CPU3_IER_CLR_0

CPU3s Clear Interrupt Enable for CPU3

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU3.

Offset: 0xa0 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR

Bit	Reset	Description
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.2.34 PRI_ICTLR_CPU3_IEP_CLASS_0

CPU3s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU3. 1 = FIQ, 0 = IRQ.

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SDMMC4

Bit	Reset	Description
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
15	0x0	SDMMC2
14	0x0	SDMMC1
12	0x0	VDE_SXE
11	0x0	VDE_BSEA
10	0x0	VDE_BSEV
9	0x0	VDE_SYNC_TOKEN
8	0x0	VDE_UCQ
7	0x0	SHR_SEM_OUTBOX_EMPTY
6	0x0	SHR_SEM_OUTBOX_FULL
5	0x0	SHR_SEM_INBOX_EMPTY
4	0x0	SHR_SEM_INBOX_FULL
3	0x0	CEC
2	0x0	RTC
1	0x0	TMR2
0	0x0	TMR1

3.2.3 Arb_gnt Specific Interrupt Controller Registers

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

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

The Arbitration Semaphores can also generate an interrupt when a hardware resource becomes available. The registers in this module configure these interrupts. When a 1 is set in the corresponding bit position of the Arbitration Semaphore Interrupt

Source Register (CPU_enable or COP_enable), an interrupt will be generated when the processor achieves Grant Status for that resource.

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

3.2.3.1 PRI_ICTLR_ARBGNT_CPU_STATUS_0

CPU Arbitration Semaphore Interrupt Status Register

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

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

3.2.3.2 PRI_ICTLR_ARBGNT_CPU_ENABLE_0

CPU Arbitration Semaphore Interrupt Enable Register

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

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

3.2.3.3 PRI_ICTLR_ARBGNT_COP_STATUS_0

COP Arbitration Semaphore Interrupt Status Register

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

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

3.2.3.4 PRI_ICTLR_ARBGNT_COP_ENABLE_0

COP Arbitration Semaphore Interrupt Enable Register

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

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

3.2.4 Second Interrupt Controller Registers

3.2.4.1 SEC_ICTLR_VIRQ_CPU_0

Valid Interrupt Request Status for CPU Register

Flags set by hardware, cleared by software.

Offset: 0x100 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4

Bit	Reset	Description
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.2 SEC_ICTLR_VIRQ_COP_0

Valid Interrupt Status for COP Register

Flags set by hardware, cleared by software.

Offset: 0x104 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT

Bit	Reset	Description
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.3 SEC_ICTLR_VFIQ_CPU_0

FIQ Valid Interrupt Status for CPU Register

Flags set by hardware, cleared by software.

Offset: 0x108 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME

Bit	Reset	Description
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.4 SEC_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for COP Register

Flags set by hardware, cleared by software.

Offset: 0x10c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI

Bit	Reset	Description
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.5 SEC_ICTLR_ISR_0

Latched Interrupt Status Register (HW)

Set by hardware event, cleared at source by software.

Offset: 0x110 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP

Bit	Reset	Description
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.6 SEC_ICTLR_FIR_0

Forced Interrupt Status Register (SW)

Set during write to FIR_SET, cleared during write to FIR_CLR.

Offset: 0x114 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5

Bit	Reset	Description
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.7 SEC_ICTLR_FIR_SET_0

Force Interrupt Register Set

Set Forced Interrupt Bit. Writing a 1 will set an interrupt.

Offset: 0x118 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1

Bit	Reset	Description
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.8 SEC_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

Clear Forced Interrupt Bit: Writing a 1 will clear the forced interrupt.

Offset: 0x11c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.9 SEC_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x120 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3

Bit	Reset	Description
1	X	GPIO2
0	X	GPIO1

3.2.4.10 SEC_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU.

Offset: 0x124 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C

Bit	Reset	Description
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.11 SEC_ICTLR_CPU_IER_CLR_0

CPUs Clear Interrupt Enable for CPU

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU.

Offset: 0x128 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4

Bit	Reset	Description
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.12 SEC_ICTLR_CPU_IEP_CLASS_0

CPUs Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU. 1 = FIQ, 0 = IRQ.

Offset: 0x12c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC

Bit	Reset	Description
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.13 SEC_ICTLR_COP_IER_0

Enabled Interrupt Source for COP Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x130 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC

Bit	Reset	Description
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.14 SEC_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for COP.

Offset: 0x134 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON

Bit	Reset	Description
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.15 SEC_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

Writing a 1 in any bit position will disable the corresponding Interrupt Source for COP.

Offset: 0x138 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE

Bit	Reset	Description
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.16 SEC_ICTLR_COP_IEP_CLASS_0

COPs Interrupt Enable Priority Class (FIQ/IRQ) Register

Set Priority Interrupt Source for COP. 1 = FIQ, 0 = IRQ.

Offset: 0x13c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3

Bit	Reset	Description
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.17 SEC_ICTLR_VIRQ_CPU1_0

Valid Interrupt Request Status for CPU1 Register

Flags set by hardware, cleared by software.

Offset: 0x160 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C

Bit	Reset	Description
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.18 SEC_ICTLR_VFIQ_CPU1_0

FIQ Valid Interrupt Status for CPU1 Register

Flags set by hardware, cleared by software.

Offset: 0x164 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4

Bit	Reset	Description
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.19 SEC_ICTLR_CPU1_IER_0

Enabled Interrupt Source for CPU1 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x168 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC

Bit	Reset	Description
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.20 SEC_ICTLR_CPU1_IER_SET_0

Set Interrupt Enable for CPU1 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU1.

Offset: 0x16c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC

Bit	Reset	Description
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.21 SEC_ICTLR_CPU1_IER_CLR_0

CPU1s Clear Interrupt Enable for CPU1

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU1.

Offset: 0x170 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON

Bit	Reset	Description
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.22 SEC_ICTLR_CPU1_IEP_CLASS_0

CPU1s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU1. 1 = FIQ, 0 = IRQ.

Offset: 0x174 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE

Bit	Reset	Description
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.23 SEC_ICTLR_VIRQ_CPU2_0

Valid Interrupt Request Status for CPU2 Register

Flags set by hardware, cleared by software.

Offset: 0x178 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3

Bit	Reset	Description
1	X	GPIO2
0	X	GPIO1

3.2.4.24 SEC_ICTLR_VFIQ_CPU2_0

FIQ Valid Interrupt Status for CPU2 Register

Flags set by hardware, cleared by software.

Offset: 0x17c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C

Bit	Reset	Description
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.25 SEC_ICTLR_CPU2_IER_0

Enabled Interrupt Source for CPU2 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x180 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4

Bit	Reset	Description
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.26 SEC_ICTLR_CPU2_IER_SET_0

Set Interrupt Enable for CPU2 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU2.

Offset: 0x184 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC

Bit	Reset	Description
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.27 SEC_ICTLR_CPU2_IER_CLR_0

CPU2s Clear Interrupt Enable for CPU2

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU2.

Offset: 0x188 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC

Bit	Reset	Description
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.28 SEC_ICTLR_CPU2_IEP_CLASS_0

CPU2s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU2. 1 = FIQ, 0 = IRQ.

Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON

Bit	Reset	Description
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.29 SEC_ICTLR_VIRQ_CPU3_0

Valid Interrupt Request Status for CPU3 Register

Flags set by hardware, cleared by software.

Offset: 0x190 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE

Bit	Reset	Description
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.30 SEC_ICTLR_VFIQ_CPU3_0

FIQ Valid Interrupt Status for CPU3 Register

Flags set by hardware, cleared by software.

Offset: 0x194 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3

Bit	Reset	Description
1	X	GPIO2
0	X	GPIO1

3.2.4.31 SEC_ICTLR_CPU3_IER_0

Enabled Interrupt Source for CPU3 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x198 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	CVDVFS
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	USB3_DEV_PME
24	X	USB3_DEV_SMI
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EDP
18	X	TSEC
17	X	XUSB_PADCTL
16	X	THERMAL
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	USB3_DEV_HOST
11	X	USB3_HOST_PME
10	X	TMR4
9	X	TMR3
8	X	USB3_HOST_SMI
7	X	USB3_HOST_INT
6	X	I2C

Bit	Reset	Description
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.4.32 SEC_ICTLR_CPU3_IER_SET_0

Set Interrupt Enable for CPU3 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU3.

Offset: 0x19c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4

Bit	Reset	Description
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.33 SEC_ICTLR_CPU3_IER_CLR_0

CPU3s Clear Interrupt Enable for CPU3

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU3.

Offset: 0x1a0 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC

Bit	Reset	Description
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.4.34 SEC_ICTLR_CPU3_IEP_CLASS_0

CPU3s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU3. 1 = FIQ, 0 = IRQ.

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	CVDVFS
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	USB3_DEV_PME
24	0x0	USB3_DEV_SMI
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EDP
18	0x0	TSEC

Bit	Reset	Description
17	0x0	XUSB_PADCTL
16	0x0	THERMAL
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	USB3_DEV_HOST
11	0x0	USB3_HOST_PME
10	0x0	TMR4
9	0x0	TMR3
8	0x0	USB3_HOST_SMI
7	0x0	USB3_HOST_INT
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.5 Third Interrupt Controller Registers

3.2.5.1 TRI_ICTLR_VIRQ_CPU_0

Valid Interrupt Request Status for CPU Register

Flags set by hardware, cleared by software.

Offset: 0x200 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT

Bit	Reset	Description
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.2 TRI_ICTLR_VIRQ_COP_0

Valid Interrupt Status for COP Register

Flags set by hardware, cleared by software.

Offset: 0x204 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6

Bit	Reset	Description
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.3 TRI_ICTLR_VFIQ_CPU_0

FIQ Valid Interrupt Status for CPU Register

Flags set by hardware, cleared by software.

Offset: 0x208 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7

Bit	Reset	Description
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.4 TRI_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for COP Register

Flags set by hardware, cleared by software.

Offset: 0x20c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD

Bit	Reset	Description
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.5 TRI_ICTLR_ISR_0

Latched Interrupt Status Register (HW)

Set by hardware event, cleared at source by software.

Offset: 0x210 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3

Bit	Reset	Description
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.6 TRI_ICTLR_FIR_0

Forced Interrupt Status Register (SW)

Set during write to FIR_SET, cleared during write to FIR_CLR.

Offset: 0x214 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4

Bit	Reset	Description
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.7 TRI_ICTLR_FIR_SET_0

Force Interrupt Register Set

Set Forced Interrupt Bit. Writing a 1 will set an interrupt.

Offset: 0x218 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5

Bit	Reset	Description
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.8 TRI_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

Clear Forced Interrupt Bit: Writing a 1 will clear the forced interrupt.

Offset: 0x21c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.9 TRI_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x220 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.10 TRI_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU.

Offset: 0x224 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.11 TRI_ICTLR_CPU_IER_CLR_0

CPUs Clear Interrupt Enable for CPU

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU.

Offset: 0x228 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.12 TRI_ICTLR_CPU_IEP_CLASS_0

CPUs Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU. 1 = FIQ, 0 = IRQ.

Offset: 0x22c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.13 TRI_ICTLR_COP_IER_0

Enabled Interrupt Source for COP Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x230 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.14 TRI_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for COP.

Offset: 0x234 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.15 TRI_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

Writing a 1 in any bit position will disable the corresponding Interrupt Source for COP.

Offset: 0x238 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.16 TRI_ICTLR_COP_IEP_CLASS_0

COPs Interrupt Enable Priority Class (FIQ/IRQ) Register

Set Priority Interrupt Source for COP. 1 = FIQ, 0 = IRQ.

Offset: 0x23c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.17 TRI_ICTLR_VIRQ_CPU1_0

Valid Interrupt Request Status for CPU1 Register

Flags set by hardware, cleared by software.

Offset: 0x260 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.18 TRI_ICTLR_VFIQ_CPU1_0

FIQ Valid Interrupt Status for CPU1 Register

Flags set by hardware, cleared by software.

Offset: 0x264 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.19 TRI_ICTLR_CPU1_IER_0

Enabled Interrupt Source for CPU1 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x268 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.20 TRI_ICTLR_CPU1_IER_SET_0

Set Interrupt Enable for CPU1 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU1.

Offset: 0x26c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.21 TRI_ICTLR_CPU1_IER_CLR_0

CPU1s Clear Interrupt Enable for CPU1

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU1.

Offset: 0x270 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.22 TRI_ICTLR_CPU1_IEP_CLASS_0

CPU1s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source For CPU1. 1 = FIQ, 0 = IRQ.

Offset: 0x274 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.23 TRI_ICTLR_VIRQ_CPU2_0

Valid Interrupt Request Status for CPU2 Register

Flags set by hardware, cleared by software.

Offset: 0x278 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.24 TRI_ICTLR_VFIQ_CPU2_0

FIQ Valid Interrupt Status for CPU2 Register

Flags set by hardware, cleared by software.

Offset: 0x27c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.25 TRI_ICTLR_CPU2_IER_0

Enabled Interrupt Source for CPU2 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x280 | Read/Write: RO | Reset: xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.26 TRI_ICTLR_CPU2_IER_SET_0

Set Interrupt Enable for CPU2 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU2.

Offset: 0x284 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.27 TRI_ICTLR_CPU2_IER_CLR_0

CPU2s Clear Interrupt Enable for CPU2

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU2.

Offset: 0x288 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.28 TRI_ICTLR_CPU2_IEP_CLASS_0

CPU2s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU2. 1 = FIQ, 0 = IRQ.

Offset: 0x28c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.29 TRI_ICTLR_VIRQ_CPU3_0

Valid Interrupt Request Status for CPU3 Register

Flags set by hardware, cleared by software.

Offset: 0x290 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.30 TRI_ICTLR_VFIQ_CPU3_0

FIQ Valid Interrupt Status for CPU3 Register

Flags set by hardware, cleared by software.

Offset: 0x294 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.31 TRI_ICTLR_CPU3_IER_0

Enabled Interrupt Source for CPU3 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x298 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
26	X	UARTD
25	X	GPIO7
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
15	X	SBC6
14	X	EMC
13	X	MC
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MSENC
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.5.32 TRI_ICTLR_CPU3_IER_SET_0

Set Interrupt Enable for CPU3 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU3.

Offset: 0x29c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.33 TRI_ICTLR_CPU3_IER_CLR_0

CPU3s Clear Interrupt Enable for CPU3

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU3.

Offset: 0x2a0 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5.34 TRI_ICTLR_CPU3_IEP_CLASS_0

CPU3s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU3. 1 = FIQ, 0 = IRQ.

Offset: 0x2a4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
26	0x0	UARTD
25	0x0	GPIO7
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MSENC
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.6 Fourth Interrupt Controller Registers

3.2.6.1 QUAD_ICTLR_VIRQ_CPU_0

Valid Interrupt Request Status for CPU Register

Flags set by hardware, cleared by software

Offset: 0x300 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.2 QUAD_ICTLR_VIRQ_COP_0

Valid Interrupt Status for COP Register

Flags set by hardware, cleared by software.

Offset: 0x304 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.3 QUAD_ICTLR_VFIQ_CPU_0

FIQ Valid Interrupt Status for CPU Register

Flags set by hardware, cleared by software.

Offset: 0x308 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.4 QUAD_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for COP Register

Flags set by hardware, cleared by software.

Offset: 0x30c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.5 QUAD_ICTLR_ISR_0

Latched Interrupt Status Register (HW)

Set by hardware event, cleared at source by software.

Offset: 0x310 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.6 QUAD_ICTLR_FIR_0

Forced Interrupt Status Register (SW)

Set during write to FIR_SET, cleared during write to FIR_CLR.

Offset: 0x314 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.7 QUAD_ICTLR_FIR_SET_0

Force Interrupt Register Set

Set Forced Interrupt Bit. Writing a 1 will set an interrupt.

Offset: 0x318 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.8 QUAD_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

Clear Forced Interrupt Bit: Writing a 1 will clear the forced interrupt.

Offset: 0x31c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.9 QUAD_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x320 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.10 QUAD_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU.

Offset: 0x324 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.11 QUAD_ICTLR_CPU_IER_CLR_0

CPUs Clear Interrupt Enable for CPU

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU.

Offset: 0x328 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.12 QUAD_ICTLR_CPU_IEP_CLASS_0

CPUs Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU. 1 = FIQ, 0 = IRQ.

Offset: 0x32c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.13 QUAD_ICTLR_COP_IER_0

Enabled Interrupt Source for COP Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x330 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.14 QUAD_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for COP.

Offset: 0x334 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.15 QUAD_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

Writing a 1 in any bit position will disable the corresponding Interrupt Source for COP.

Offset: 0x338 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.16 QUAD_ICTLR_COP_IEP_CLASS_0

COPs Interrupt Enable Priority Class (FIQ/IRQ) Register

Set Priority Interrupt Source for COP. 1 = FIQ, 0 = IRQ.

Offset: 0x33c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.17 QUAD_ICTLR_VIRQ_CPU1_0

Valid Interrupt Request Status for CPU1 Register

Flags set by hardware, cleared by software.

Offset: 0x360 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.18 QUAD_ICTLR_VFIQ_CPU1_0

FIQ Valid Interrupt Status for CPU1 Register

Flags set by hardware, cleared by software.

Offset: 0x364 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.19 QUAD_ICTLR_CPU1_IER_0

Enabled Interrupt Source for CPU1 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x368 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.20 QUAD_ICTLR_CPU1_IER_SET_0

Set Interrupt Enable for CPU1 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU1.

Offset: 0x36c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.21 QUAD_ICTLR_CPU1_IER_CLR_0

CPU1s Clear Interrupt Enable for CPU1

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU1.

Offset: 0x370 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.22 QUAD_ICTLR_CPU1_IEP_CLASS_0

CPU1s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU1. 1 = FIQ, 0 = IRQ.

Offset: 0x374 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.23 QUAD_ICTLR_VIRQ_CPU2_0

Valid Interrupt Request Status for CPU2 Register

Flags set by hardware, cleared by software.

Offset: 0x378 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.24 QUAD_ICTLR_VFIQ_CPU2_0

FIQ Valid Interrupt Status for CPU2 Register

Flags set by hardware, cleared by software.

Offset: 0x37c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.25 QUAD_ICTLR_CPU2_IER_0

Enabled Interrupt Source for CPU2 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x380 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.26 QUAD_ICTLR_CPU2_IER_SET_0

Set Interrupt Enable for CPU2 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU2.

Offset: 0x384 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.27 QUAD_ICTLR_CPU2_IER_CLR_0

CPU2s Clear Interrupt Enable for CPU2

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU2.

Offset: 0x388 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.28 QUAD_ICTLR_CPU2_IEP_CLASS_0

CPU2s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU2. 1 = FIQ, 0 = IRQ.

Offset: 0x38c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.29 QUAD_ICTLR_VIRQ_CPU3_0

Valid Interrupt Request Status for CPU3 Register

Flags set by hardware, cleared by software.

Offset: 0x390 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.30 QUAD_ICTLR_VFIQ_CPU3_0

FIQ Valid Interrupt Status for CPU3 Register

Flags set by hardware, cleared by software.

Offset: 0x394 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.31 QUAD_ICTLR_CPU3_IER_0

Enabled Interrupt Source for CPU3 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x398 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	HIER_GROUP1_CPU
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	HIER_GROUP1_COP
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13
20	X	APB_DMA_CH12
19	X	APB_DMA_CH11
18	X	APB_DMA_CH10
17	X	APB_DMA_CH9
16	X	APB_DMA_CH8
15	X	APB_DMA_CH7
14	X	APB_DMA_CH6
13	X	APB_DMA_CH5
12	X	APB_DMA_CH4
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
5	X	AVP_CACHE
1	X	USB3
0	X	SNOR

3.2.6.32 QUAD_ICTLR_CPU3_IER_SET_0

Set Interrupt Enable for CPU3 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU3.

Offset: 0x39c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.33 QUAD_ICTLR_CPU3_IER_CLR_0

CPU3s Clear Interrupt Enable for CPU3

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU3.

Offset: 0x3a0 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.6.34 QUAD_ICTLR_CPU3_IEP_CLASS_0

CPU3s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU3. 1 = FIQ, 0 = IRQ.

Offset: 0x3a4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	HIER_GROUP1_CPU
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	HIER_GROUP1_COP
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15
22	0x0	APB_DMA_CH14
21	0x0	APB_DMA_CH13
20	0x0	APB_DMA_CH12
19	0x0	APB_DMA_CH11
18	0x0	APB_DMA_CH10
17	0x0	APB_DMA_CH9
16	0x0	APB_DMA_CH8
15	0x0	APB_DMA_CH7
14	0x0	APB_DMA_CH6
13	0x0	APB_DMA_CH5
12	0x0	APB_DMA_CH4
11	0x0	APB_DMA_CH3
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
5	0x0	AVP_CACHE
1	0x0	USB3
0	0x0	SNOR

3.2.7 Fifth Interrupt Controller Registers

3.2.7.1 PENTA_ICTLR_VIRQ_CPU_0

Valid Interrupt Request Status for CPU Register

Flags set by hardware, cleared by software.

Offset: 0x400 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19

Bit	Reset	Description
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.2 PENTA_ICTLR_VIRQ_COP_0

Valid Interrupt Status for COP Register

Flags set by hardware, cleared by software.

Offset: 0x404 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23

Bit	Reset	Description
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.3 PENTA_ICTLR_VFIQ_CPU_0

FIQ Valid Interrupt Status for CPU Register

Flags set by hardware, cleared by software.

Offset: 0x408 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27

Bit	Reset	Description
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.4 PENTA_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for COP Register

Flags set by hardware, cleared by software.

Offset: 0x40c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31

Bit	Reset	Description
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.5 PENTA_ICTLR_ISR_0

Latched Interrupt Status Register (HW)

Set by hardware event, cleared at source by software.

Offset: 0x410 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR

Bit	Reset	Description
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.6 PENTA_ICTLR_FIR_0

Forced Interrupt Status Register (SW)

Set during write to FIR_SET, cleared during write to FIR_CLR.

Offset: 0x414 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS

Bit	Reset	Description
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.7 PENTA_ICTLR_FIR_SET_0

Force Interrupt Register Set

Set Forced Interrupt Bit. Writing a 1 will set an interrupt.

Offset: 0x418 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9

Bit	Reset	Description
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.8 PENTA_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

Clear Forced Interrupt Bit: Writing a 1 will clear the forced interrupt.

Offset: 0x41c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18

Bit	Reset	Description
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.9 PENTA_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x420 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22

Bit	Reset	Description
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.10 PENTA_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU.

Offset: 0x424 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26

Bit	Reset	Description
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.11 PENTA_ICTLR_CPU_IER_CLR_0

CPUs Clear Interrupt Enable for CPU

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU.

Offset: 0x428 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30

Bit	Reset	Description
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.12 PENTA_ICTLR_CPU_IEP_CLASS_0

CPUs Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source For CPU. 1 = FIQ, 0 = IRQ.

Offset: 0x42c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR

Bit	Reset	Description
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.13 PENTA_ICTLR_COP_IER_0

Enabled Interrupt Source for COP Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x430 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS

Bit	Reset	Description
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.14 PENTA_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for COP.

Offset: 0x434 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8

Bit	Reset	Description
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.15 PENTA_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

Writing a 1 in any bit position will disable the corresponding Interrupt Source for COP.

Offset: 0x438 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18

Bit	Reset	Description
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.16 PENTA_ICTLR_COP_IEP_CLASS_0

COPs Interrupt Enable Priority Class (FIQ/IRQ) Register

Set Priority Interrupt Source For COP. 1 = FIQ, 0 = IRQ.

Offset: 0x43c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22

Bit	Reset	Description
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.17 PENTA_ICTLR_VIRQ_CPU1_0

Valid Interrupt Request Status for CPU1 Register

Flags set by hardware, cleared by software.

Offset: 0x460 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26

Bit	Reset	Description
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.18 PENTA_ICTLR_VFIQ_CPU1_0

FIQ Valid Interrupt Status for CPU1 Register

Flags set by hardware, cleared by software.

Offset: 0x464 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30

Bit	Reset	Description
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.19 PENTA_ICTLR_CPU1_IER_0

Enabled Interrupt Source for CPU1 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x468 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR

Bit	Reset	Description
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.20 PENTA_ICTLR_CPU1_IER_SET_0

Set Interrupt Enable for CPU1 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU1.

Offset: 0x46c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS

Bit	Reset	Description
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.21 PENTA_ICTLR_CPU1_IER_CLR_0

CPU1s Clear Interrupt Enable for CPU1

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU1.

Offset: 0x470 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8

Bit	Reset	Description
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.22 PENTA_ICTLR_CPU1_IEP_CLASS_0

CPU1s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source for CPU1. 1 = FIQ, 0 = IRQ.

Offset: 0x474 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18

Bit	Reset	Description
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.23 PENTA_ICTLR_VIRQ_CPU2_0

Valid Interrupt Request Status for CPU2 Register

Flags set by hardware, cleared by software.

Offset: 0x478 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22

Bit	Reset	Description
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.24 PENTA_ICTLR_VFIQ_CPU2_0

FIQ Valid Interrupt Status for CPU2 Register

Flags set by hardware, cleared by software.

Offset: 0x47c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26

Bit	Reset	Description
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.25 PENTA_ICTLR_CPU2_IER_0

Enabled Interrupt Source for CPU2 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x480 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30

Bit	Reset	Description
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.26 PENTA_ICTLR_CPU2_IER_SET_0

Set Interrupt Enable for CPU2 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU2.

Offset: 0x484 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR

Bit	Reset	Description
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.27 PENTA_ICTLR_CPU2_IER_CLR_0

CPU2s Clear Interrupt Enable for CPU2

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU2.

Offset: 0x488 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS

Bit	Reset	Description
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.28 PENTA_ICTLR_CPU2_IEP_CLASS_0

CPU2s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source For CPU2. 1 = FIQ, 0 = IRQ.

Offset: 0x48c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8

Bit	Reset	Description
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.29 PENTA_ICTLR_VIRQ_CPU3_0

Valid Interrupt Request Status for CPU3 Register

Flags set by hardware, cleared by software.

Offset: 0x490 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18

Bit	Reset	Description
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.30 PENTA_ICTLR_VFIQ_CPU3_0

FIQ Valid Interrupt Status for CPU3 Register

Flags set by hardware, cleared by software

Offset: 0x494 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22

Bit	Reset	Description
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.31 PENTA_ICTLR_CPU3_IER_0

Enabled Interrupt Source for CPU3 Register

Interrupt Enable Status. 0 = Disabled.

Offset: 0x498 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	EMC0
30	X	MC0
28	X	TMR0
27	X	TMR9
26	X	TMR8
25	X	TMR7
24	X	TMR6
23	X	SDMMC4_SYS
22	X	SDMMC3_SYS
21	X	SDMMC2_SYS
20	X	SDMMC1_SYS
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR
15	X	APB_DMA_CH31
14	X	APB_DMA_CH30
13	X	APB_DMA_CH29
12	X	APB_DMA_CH28
11	X	APB_DMA_CH27
10	X	APB_DMA_CH26

Bit	Reset	Description
9	X	APB_DMA_CH25
8	X	APB_DMA_CH24
7	X	APB_DMA_CH23
6	X	APB_DMA_CH22
5	X	APB_DMA_CH21
4	X	APB_DMA_CH20
3	X	APB_DMA_CH19
2	X	APB_DMA_CH18
1	X	APB_DMA_CH17
0	X	APB_DMA_CH16

3.2.7.32 PENTA_ICTLR_CPU3_IER_SET_0

Set Interrupt Enable for CPU3 Register

Writing a 1 in any bit position will enable the corresponding Interrupt Source for CPU3.

Offset: 0x49c | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30

Bit	Reset	Description
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.33 PENTA_ICTLR_CPU3_IER_CLR_0

CPU3s Clear Interrupt Enable for CPU3

Writing a 1 in any bit position will disable the corresponding Interrupt Source for CPU3.

Offset: 0x4a0 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR

Bit	Reset	Description
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.7.34 PENTA_ICTLR_CPU3_IEP_CLASS_0

CPU3s Interrupt Enable Priority Class (FIQ/IRQ)

Set Priority Interrupt Source For CPU3. 1 = FIQ, 0 = IRQ.

Offset: 0x4a4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC0
30	0x0	MC0
28	0x0	TMR0
27	0x0	TMR9
26	0x0	TMR8
25	0x0	TMR7
24	0x0	TMR6
23	0x0	SDMMC4_SYS
22	0x0	SDMMC3_SYS

Bit	Reset	Description
21	0x0	SDMMC2_SYS
20	0x0	SDMMC1_SYS
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR
15	0x0	APB_DMA_CH31
14	0x0	APB_DMA_CH30
13	0x0	APB_DMA_CH29
12	0x0	APB_DMA_CH28
11	0x0	APB_DMA_CH27
10	0x0	APB_DMA_CH26
9	0x0	APB_DMA_CH25
8	0x0	APB_DMA_CH24
7	0x0	APB_DMA_CH23
6	0x0	APB_DMA_CH22
5	0x0	APB_DMA_CH21
4	0x0	APB_DMA_CH20
3	0x0	APB_DMA_CH19
2	0x0	APB_DMA_CH18
1	0x0	APB_DMA_CH17
0	0x0	APB_DMA_CH16

3.2.8 Hier Group Interrupts

Registers for Hier group interrupts for CPU and COP.

3.2.8.1 HIER_GROUP_CPU_ENABLE_0

Hier Group Enable for CPU

Offset: 0x800 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CPU_ENABLE: Enable for the hier group interrupt for CPU

3.2.8.2 HIER_GROUP_CPU_STATUS_0

Hier Group Status (ISR) Register for CPU

Offset: 0x804 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CPU_STATUS: Status for the hier group interrupt for CPU

3.2.8.3 HIER_GROUP_COP_ENABLE_0

Hier Group Enable for COP

Offset: 0x808 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COP_ENABLE: Enable for the hier group interrupt for COP

3.2.8.4 HIER_GROUP_COP_STATUS_0

Hier Group Status (ISR) for COP

Offset: 0x80c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COP_STATUS: Status for the hier group interrupt for COP

3.2.8.5 HIER_GROUP_FIR_STATUS_0

Status Register for FIR Register in Hier Group

Offset: 0x810 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	FIR_STATUS: Software interrupts for hier group

3.2.8.6 HIER_GROUP_FIR_SET_0

FIR Set for Hier Group

Offset: 0x814 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	FIR_SET:FIR set for hier group

3.2.8.7 HIER_GROUP_FIR_CLEAR_0

FIR Clear for Hier Group

Offset: 0x818 | Read/Write: WO | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	FIR_CLEAR: FIR clear for hier group

4.0 ARBITRATION SEMAPHORES

4.1 Overview

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

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

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

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

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

4.2 Semaphore Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

4.2.1 ARB_SEMA_SMP_GNT_ST_0

Semaphore Granted Status Register

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

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

4.2.2 ARB_SEMA_SMP_GET_0

Request Arbitration Semaphore Register

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

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

4.2.3 ARB_SEMA_SMP_PUT_0

Arbitration Semaphore Put Request Register

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

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

4.2.4 ARB_SEMA_SMP_REQ_ST_0

Arbitration Request Pending Status Register

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

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

5.0 ATOMICS

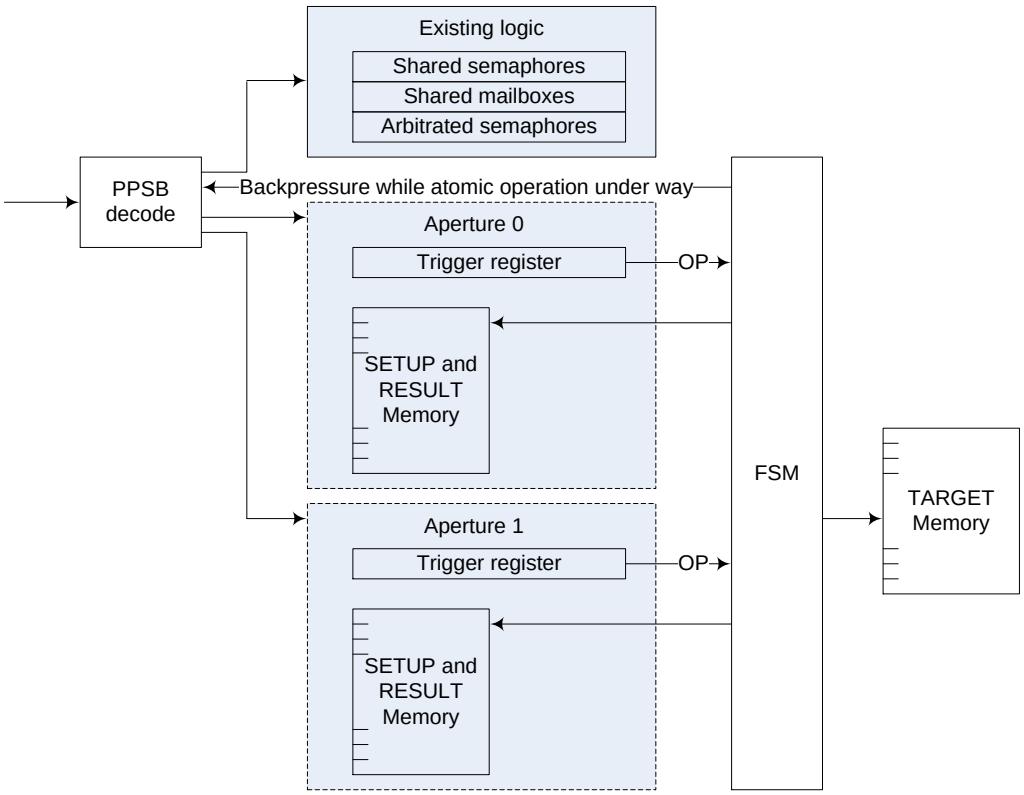
5.1 Introduction

The Atomics block assists software tasks with maintaining synchronization. It is parallel to the existing Semaphores block, and extends that block's functionality with new capabilities.

5.2 Functionality

The figure below shows a reference decomposition of the hardware synchronization elements in functional blocks.

Figure 4: Reference Block Diagram



5.3 Synchronization Elements

The synchronization elements are atomic primitives. Their main advantage is to allow some specific algorithms that access shared resources to be performed lock free, for better performance.

In general, the atomic primitives below have results equivalent to atomic RMW operations on a target register. The cycle is made atomic without using locked bus accesses as explained below.

5.3.1 General Principles

Atomic primitives maintain atomicity by splitting the whole operation in three phases:

- A setup phase where parameters are prepared in registers. There are as many sets of setup registers as there are apertures.

- A trigger register that starts a specific operation. The trigger register fits in one 32-bit word to ensure atomic issue of the operation. There is one trigger register per aperture. The trigger register specifies:
  - The primitive operation to be performed.
  - The index of the target register for the primitive operation.
- A result phase where the results of the operation are made available. There are as many sets of result registers as there are sets of setup registers. The operation, once triggered, is executed atomically.

The setup, result, and trigger registers are duplicated in separate apertures, one aperture per client of the block. There are two such apertures in Tegra® 4 processors, normally one used by CPU, the other by AVP, but this is not enforced by hardware as it is for the arbitrated semaphores.

Setup, target, and result registers are 32 bits each with no further structure imposed by hardware. The trigger register is 32 bits, structured in command and target fields.

The description of the different operations is based on pseudo code that uses the following conventions:

- ATOMIC_OP identifies the exact operation and index ID identifies a given target register. Both ATOMIC_OP and ID are duplicated per aperture
- Index A identifies the aperture from which the operation is executed.
- The setup registers in a set are stored in arrays v and c. The result registers are stored in array old[1]. All of these are duplicated per aperture.
- The target registers are not duplicated per aperture and are stored in an array T.
- The semantic of assignment is like C. All assignments are assumed to take place instantaneously. Intermediate values are used to get the equivalent of clocked operations of the hardware. Typically, the hardware implementation has no intermediate values if operations are performed in one clock cycle.
- Arithmetic operations are performed as signed 32-bit operation modulo 2^{32} using 2's complement coding.

5.3.2 Atomic Exchange

In an atomic exchange, its only purpose is to be able to read a value and replace it with another value in an atomic fashion.

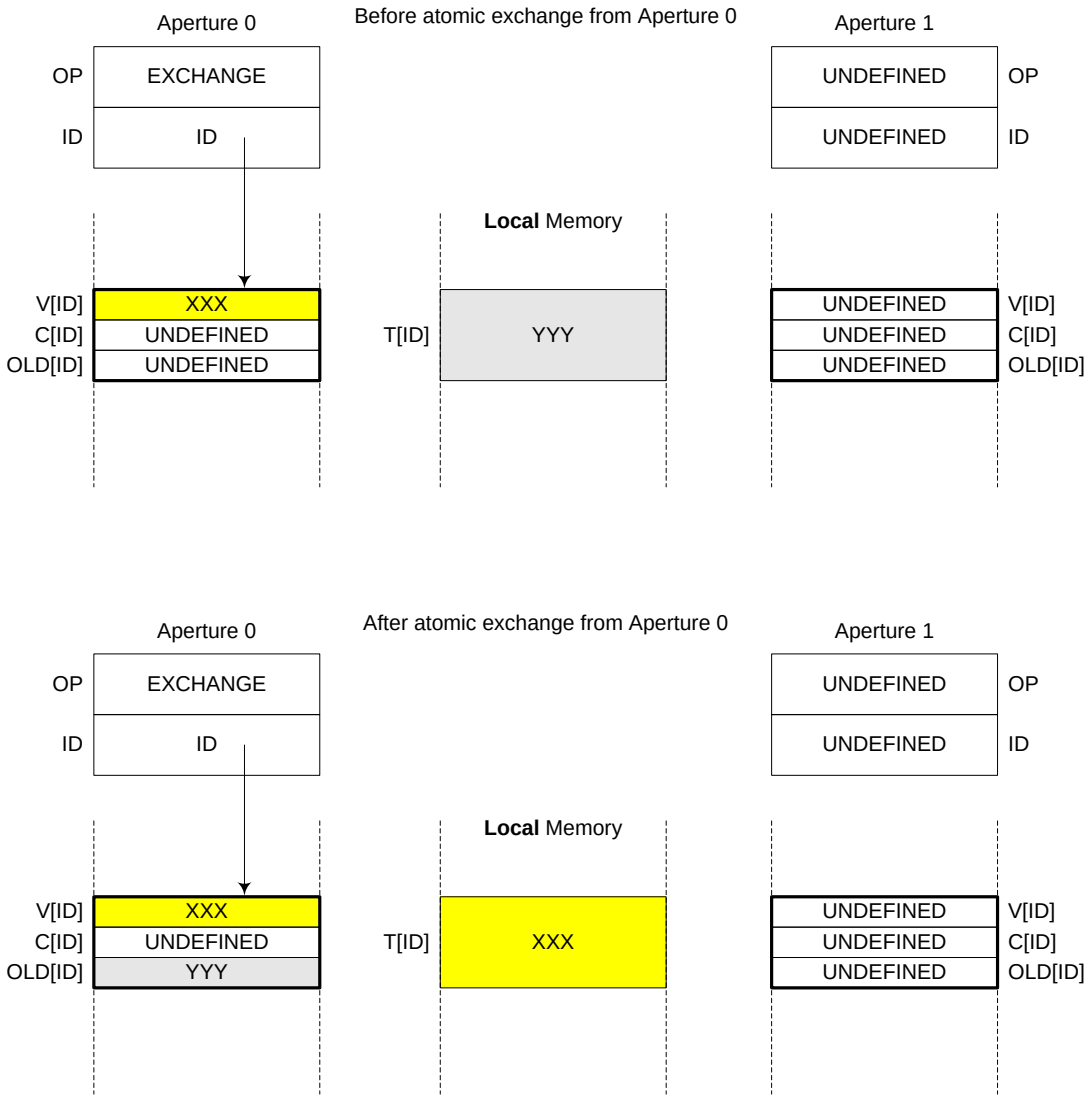
The corresponding pseudo code is

```
if (ATOMIC_OP[A] == EXCHANGE) {
  old[A][ID[A]] = T[ID[A]] ;
  T[ID[A]] = v[A][ID[A]]
```

The figure below illustrates an atomic exchange performed using aperture 0.

[1] The result register always contains the old value of the target register, i.e. the value before applying the atomic operation. This is true even for PUT, so that PUT and EXCHANGE are semantically equivalent with the current definition.

Figure 5: Atomic Exchange Operation



5.3.3 Atomic Compare and Exchange

This is a variant of atomic exchange where the exchange only takes place if the old value matches the second setup register.

```
if (ATOMIC OP[A] == COMPARE AND EXCHANGE) && (T[ID[A]] == c[A][ID[A}]) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] = v[A][ID[A]]
}
```

5.3.4 Atomic Increment

The atomic increment simply ensures that different processors can increment a shared value without losing any increment. The operation is labeled INCREMENT, but, for 32-bit operations, incrementing by 2^{32} – k is equivalent to decrementing by k.

```
if (ATOMIC_OP[A] == INCREMENT) {
  old[A][ID[A]] = Target[ID[A]]
  T[ID[A]] += v[A][ID[A]]
}
```

The variant DECREMENT_WITH_ZERO_SATURATE is similar, adding a saturation to zero when going down. This may be used to manage shared resources that are present in finite numbers.

```
if (ATOMIC_OP[A] == DECREMENT_WITH_ZERO_SATURATE) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] -= v[A][ID[A]]
  if MSB(T[ID[A]]) == 1b && MSB(r[A][ID[A]]) == 0b { T[ID[A]] = 0 ; }
}
```

5.3.5 Atomic Get and Put

These are trivial for hardware if there is only one access port, in which case they act as normal register operations. They are nevertheless defined to allow more complex future implementations where the atomic operation blocks could present multiple hardware interfaces to the rest of the system.

```
if (ATOMIC OP[A] == GET) {
  old[A][ID[A]] = T[ID[A]]
}

if (ATOMIC OP[A] == PUT) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] = v[A][ID[A]]
}
```

5.3.6 Atomic Test and Set, Test and Clear, Test and Invert

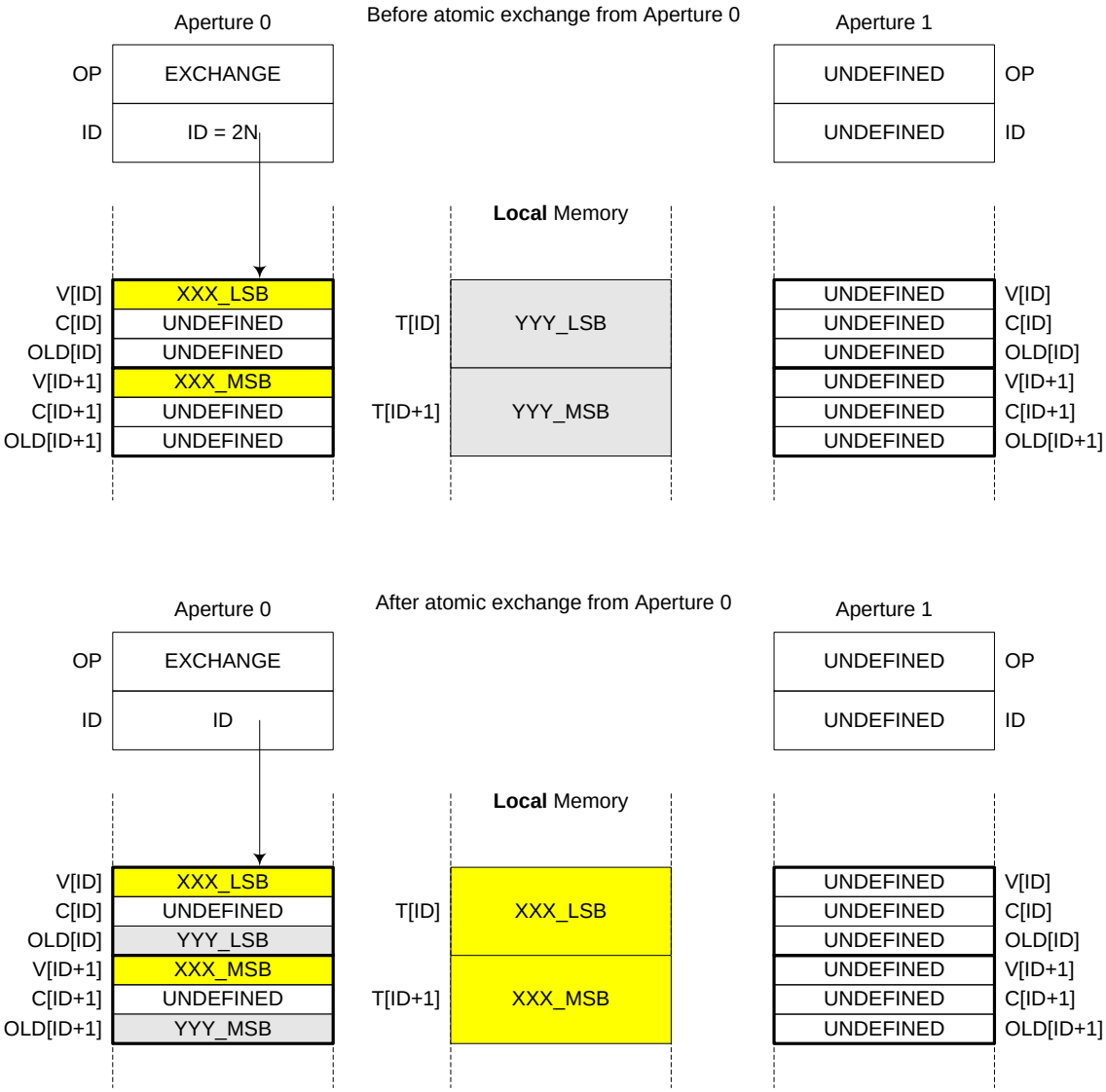
These are simple extensions of previous cases, where the operation is either a bit set, bit clear, or bit invert.

```
if (ATOMIC_OP[A] == TEST_AND_SET) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] |= v[A][ID[A]]
}

if (ATOMIC_OP[A] == TEST_AND_CLEAR) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] &= ~v[A][ID[A]]
}

if (ATOMIC_OP[A] == TEST_AND_INVERT) {
  old[A][ID[A]] = T[ID[A]]
  T[ID[A]] ^= v[A][ID[A]]
}
```

5.3.7 Width of Operations

All operations can operate on 32 bits or 64 bits. 64-bit operations use an even/odd pair of address, with the MSB at the odd address. The next figure shows how a 64-bit exchange operates. Other operations behave similarly. In the case of atomic increment and decrement operations, the carry will correctly propagate from the LSB to the MSB.

Figure 6: 64-bit Atomic Exchange Operation



5.4 Atomics Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

5.4.1 ATOMICS_AP0_TRIGGER_0

Trigger Registers

Offset: 0x0 | Read/Write: R/W | Reset: 0x00XX00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
22:16	X	ID
4	X	WIDTH64

Bit	Reset	Description
3:0	X	CMD: 0 = EXCHANGE 1 = COMPARE_AND_EXCHANGE 2 = INCREMENT 3 = DECREMENT_W_ZERO_SATURATE 4 = GET 5 = PUT 6 = TEST_AND_SET 7 = TEST_AND_CLEAR 8 = TEST_AND_INVERT

5.4.2 ATOMICS_AP1_TRIGGER_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00XX00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
22:16	X	ID
4	X	WIDTH64
3:0	X	CMD: 0 = EXCHANGE 1 = COMPARE_AND_EXCHANGE 2 = INCREMENT 3 = DECREMENT_W_ZERO_SATURATE 4 = GET 5 = PUT 6 = TEST_AND_SET 7 = TEST_AND_CLEAR 8 = TEST_AND_INVERT

5.4.3 ATOMICS_AP0_SETUP_V_0

Aperture 0

This is an array of 128 identical register entries; the register fields below apply to each entry.

Offset: 0x400..0x5ff | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	V

5.4.4 ATOMICS_AP0_SETUP_C_0

This is an array of 128 identical register entries; the register fields below apply to each entry.

Offset: 0x800..0x9ff | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C

5.4.5 ATOMICS_AP0_RESULT_0

This is an array of 128 identical register entries; the register fields below apply to each entry.

Offset: 0xc00..0xdff | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	R

5.4.6 ATOMICS_AP1_SETUP_V_0

Aperture 1

This is an array of 128 identical register entries; the register fields below apply to each entry.

Offset: 0x1000..0x11ff | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	V

5.4.7 ATOMICS_AP1_SETUP_C_0

This is an array of 128 identical register entries; the register fields below apply to each entry.

Offset: 0x1400..0x15ff | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C

5.4.8 ATOMICS_AP1_RESULT_0

This is an array of 128 identical register entries; the register fields below apply to each entry.

Offset: 0x1800..0x19ff | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	R

[THIS PAGE INTENTIONALLY LEFT BLANK]

6.0 CLOCK AND RESET CONTROLLER

The Clock and Reset (CAR) block contains all the logic needed to control most of the clocks and resets to the Tegra® 4 device. The CAR block provides the registers to program the PLLs and controls most of the clock source programming, and most of the clock dividers.

Power on reset (SYS_RESET_N_) is the primary reset for the Tegra 4 chip and is provided by the external PMC.

6.1 Hardware Features

6.1.1 External Clock Sources

The CAR block takes two external clock sources as input:

- A single external 32.768 kHz clock, normally provided by the Power Manager Integrated Circuit (PMIC)
- An oscillator clock at 12, 13, 16.8, 19.2, 26, 38.4, or 48 MHz, provided by an external crystal. Not all these frequencies may be supported by software.

6.1.2 Clock Sources and PLLs

Clock sources are provided by 12 PLLs in the Tegra 4 clock system:

- PLLM: Clock source for EMC 2x clock
- PLLX: Clock source for the fast CPU cluster and the shadow CPU
- PLLC: Clock source for general use
- PLLC2 and PLLC3: Clock source for engine scaling
- PLLP: Clock source for most peripherals
- PLLA: Audio clock sources:
  - 11.2896 MHz
  - 12.288 MHz
  - 24.576 MHz
- PLLU: Clock source for USB PHY, provides 12/60/480 MHz
- PLLD and PLLD2: Clock sources for the DSI and display subsystem
- refPLLe and PLLE generate the 100 MHz reference clock for USB 3.0 and can generate a spread-spectrum clock

There are two clock inputs to the Tegra 4 device:

- OSC: Source from external crystal
- CLK_32K_IN: 32-kHz clock provided by the PMC

6.1.3 Clock Dividers / Skippers / Multipliers

There are a number of clock skippers, clock dividers, and clock multipliers in use in the Tegra 4 devices.

- Clock skippers:
  - An M/N clock-skipping divider that is used to generate the CPU clock and also the Tegra 4 "system clock" (sclk).
  - Clock skippers that are used to generate the "AHB clock" (hclk) and "APB clock" (pclk). One that divides down by a 2-bit unsigned value (U2).
- Clock dividers:
  - An unsigned 16-bit divider (U16) that is the divider for the I2C and UART devices.

- An unsigned 8-bit divider that provides 7 bits of mantissa and 1 bit of fraction (U7.1). Tegra 4 devices have reduced the number of divider types and in particular this U7.1 divider is the default divider for most all blocks in Tegra 4 devices.

  Note: The clock-skipping divider and the U7.1 divider when programmed with the 1b fraction do not create 50/50 duty-cycle clock waveforms and may not be suitable for all modules under all circumstances.

- Clock multiplier:
  - A times-two multiplier used to double the external reference frequency by delaying and XORing the input clock with a 4-bit programmable delayed reference clock. Care must be exercised when using this dual-frequency and widely varying duty-cycle clock.

6.1.4 Power Gating and Partitions

Power gating is used to reduce leakage of idle units. For example, if a power rail is shared by unit A and unit B, then that rail (thus units) can only be powered off when both units are idle and can accept the rail power on/off latency. However, with the help of on-die power gates, each of the units can be powered off independently without powering off the rail. This is also useful to reduce latency of powering up the unit (because rail power up latency is 1000x times that of power gates).

Tegra 4 devices use the header power-gate implementation because of its power efficiency. A power-gated partition has two VDD rails. The first one is called the real power (VDD), and the second one is called the virtual power (VVDD). There are hundreds of sleep transistors (also known as power gates) that sit between the two power rails such that VVDD can be powered on/off by controlling power gates. The logic in the power-gated partition is powered by the VVDD rail.

The Tegra 4 device uses three core power rails: VDD_CPU, VDD_SoC, and VDD_AO (always-on). The VDD_CPU power rail is used for high-performance CPUs (fast CPUs), VDD_AO is used for the PMC and some of the other always-on logic. VDD_SoC is used for the rest of the core logic (including the shadow CPU) in Tegra 4 devices.

Within each power rail, some of the partitions may be power gated (referred to as PG partitions or PG domains), and others may not be power gated (referred to as NPG or non-PG domains).

In addition to three core power rails, the Tegra 4 device uses several I/O power rails which are not described in this document.

6.1.4.1 CPU Power Rail

The CPU power rail is used to power a fast CPU cluster (cluster0). Each of the four CPUs in cluster0 is independently power-gated as a separate partition. The cluster0 non-CPU (including L2) is power gated as a separate fifth partition.

To be able to power-gate cluster0 non-CPU partition, all of the CPU partitions have to be power gated first. Also, the cluster0 L2 cache has to be flushed out to DRAM before non-CPU power-gating. The L2 flushing is a long latency event. So instead of power-gating a non-CPU, the CPU rail is powered off, which is better from leakage saving perspective but costly from a power-up latency perspective (rail power-up is much longer than power-ungating).

Tegra 4 Partition	Description
CE0	Cluster0 CPU0
CE1	Cluster0 CPU1
CE2	Cluster0 CPU2
CE3	Cluster0 CPU3
C0NC	Cluster0 Non-CPU (L2 Ctrl + L2 Data (2MB) + L2 Tags + Debug, GIC/Timers, AXD, AXICIF) also known as (FL2 and FTOP)

6.1.4.2 SoC Power Rail

The SoC power rail is used to power all of the SoC logic. The following table lists the power-gated and unpower-gated domains in the VDD_SoC power rail.

PG Domain	Description
CELP	Cluster1 CPU (Shadow CPU)
C1NC	Cluster1 Non-CPU (L2 Ctrl + L2 Data (512KB) + L2 Tags + Debug, GIC/Timers, AXD, AXICIF) also known as (SL2 and STOP)
DISP	DisplayA
DISPB	DisplayB
HEG	GR2D, GSHIM, EPP
MPE	MSENC (Video Encoder)
TD (TDA,TDB, TDC,TDD, TDE,TDF)	GPU (3D)
VDE	Video Decoder Engine
VE	VI(Video Input), CSI, ISP
XUSBA	XUSB (USB3): SuperSpeed (USB3)
XUSBB	XUSB (USB3): Device Mode (USB2/1)
XUSBC	XUSB (USB3): Host Mode
NPG	AHB Cluster, TZRAM, SE, SDMMC, NAND, CoreSight APB Cluster, Audio Cluster, Fuse AVP Cluster CAR, Host1x MCA, MCB USX (USB2,USB1) Mselect, APC

6.1.4.3 AO Power Rail (VDD_AO)

The VDD_AO power rail runs the logic that needs to be active during deep sleep (LP0). The following functions are in the VDD_AO power rail:

- KBC (KeyBoard Controller)
- PMC (Power Management Controller)
- RTC (Real-Time Clock)
- Some of the pads

6.2 Clocking Architecture

Tegra 4 clocks are classified into two categories: core clocks and I/O clocks. Core clocks drive the non-I/O modules on the Tegra 4 chip. I/O clocks are custom designed for each I/O module. Only core clocks are described here.

6.2.1 Core Clocks

The core clock structure consists of:

- Sources
- Switches

- Distribution and skew balancing
- Second-level non-functional clock gates.

Core clocks originate from external clock sources which are used by on-chip PLLs to generate a set of primary clock sources. Secondary clock sources are created by dividing down various PLL outputs. This group of external, PLL, and divided PLL clocks constitutes the clock sources.

These sources feed the clock switches. Each module has its own switch. A clock switch consists of one or more clock sources selected through a Clock Source Selection Mux and driving a clock divider. Multiple clock source options are available to the module divider to allow software to choose a clock frequency based on the unit's performance requirements and overall SOC power. Switching a clock source away from a PLL when it is locking after being reprogrammed to a new frequency may also be required. There are three types of dividers: fixed divide value, fractional divider, or clock skipping (pulse eating) divider.

6.2.2 Main Clock Sources

There are three types of clock sources:

- External clock sources
- PLLs
- Derived clock sources

Table 11: External Clocks to Tegra 4 Devices

Clock Name	Description
dbg_oscout:	This clock runs at 12MHz, 13 MHz, 19.2 MHz, 26 MHz, 16.8 MHz, 38.4 MHz, or 48 MHz.
ck32khz_IB:	This clock runs at 32.768 kHz.

Table 12: Primary Clock Sources in Tegra 4 Devices

Clock Name	Description
osc_div_clk:	This clock runs at 12 MHz, 13 MHz, 16.8 MHz, 19.2 MHz, or 26MHz.
clk_m:	This clock (with DFT control) runs at 12 MHz, 13 MHz, 16.8 MHz, 19.2 MHz, 26 MHz, 38.4 MHz, or 48 MHz.
ck32khz_IB	This clock runs at 32.768 kHz.
car_clk_m:	CAR's clock tree balanced version of clk_m.
dfllCPU_out:	This is the dfllCPU's output clock. FCPU voltage following clock.
pllX_out_g:	This is the PLLX's output clock. FCPU clock.
pllC_out:	This is the PLLC's output clock. GP clock source.
pllC2_out:	This is the PLLC2's output clock. GP clock source.
pllC3_out:	This is the PLLC3's output clock. GP clock source.
pllM_out:	This is the PLLM's output clock. EMC/MC Memory clock.
pllD_out:	This is the PLLD's output clock. Display 1 clock.
pllD2_out:	This is the PLLD2's output clock. Display 2 clocks.
pllU:	This is the PLLU's output clock. USB2 I/O.
refPLLE_out:	This is the refPLLE's output clock. Low jitter reference clock for pllE, USB.
pllE_out0:	This is the PLLE's output clock. USB3 I/O.
pllP_out:	This is the PLLP's output clock. Fixed frequency GP clock source.
clk_s:	This clock (with DFT control) runs at 32.768 kHz.
car_clk_s:	CAR's clock tree balanced version of clk_s.

Table 13: Derived Clock Sources in Tegra 4 Devices

Clock Name	Description
pllxlp:	Simple div1/div2 output clocked by “pllX_out_g”. SCPU clock.
pllC_out1:	This is the U7.1 divider output clocked by “pllC_out”. GP clock source.
pllM_out1:	This is the U7.1divider output clocked by “pllM_out”. Alternate sys_clk source.
pllD_out0:	This is the divided-by-2 from PLLD’s output clock. Display 1 clock.
pllD2_out0:	This is the divided-by-2 from PLLD2’s output clock. Display 2 clock.
pllP_out1:	This is the U7.1 divider output clocked by “pllP_out”. PLLA reference clock.
pllP_out2:	This is the U7.1 divider output clocked by “pllP_out”. GP clock source.
pllP_out3:	This is the U7.1 divider output clocked by “pllP_out”. GP clock source.
pllP_out4:	This is the U7.1 divider output clocked by “pllP_out”. GP clock source.
pllA_out0:	This is the U7.1 divider output clocked by “int_pllA_out”. Audio and GP clock.
audio_sync_clk:	This clock is muxed from 9 possible audio clock inputs.
audio_2x_sync_clk:	This is the clock doubled from audio_sync_clk.

The table below lists which clock source they are derived from and their divider size.

Table 14: Clock Sources for the Derived Clocks

	dbg_oscout	pllC_out	ck32khz_lB	pllM_out	pllM_2x_out	pllP_out	int_pllA_out	spdif_in_recovered_bclk	i2s0_sclk_in	i2s1_sclk_in	i2s2_sclk_in	i2s3_sclk_in	i2s4_sclk_in	pllA_out0	vimclk_pri_in	Have super clock divider ? (# of bits)	Have U7.1 divider? (# of bits)
pllC_out1		x															8
pllM_out1				x													8
pllP_out1						x											8
pllP_out2						x											8
pllP_out3						x											8
pllP_out4						x											8
pllA_out0							x										8
audio_2x_sync_clk								0	1	2	3	4	5	6	7	-	-

These clock sources are shown in the following two figures.

Figure 7: Clocking Sources (1 of 2)

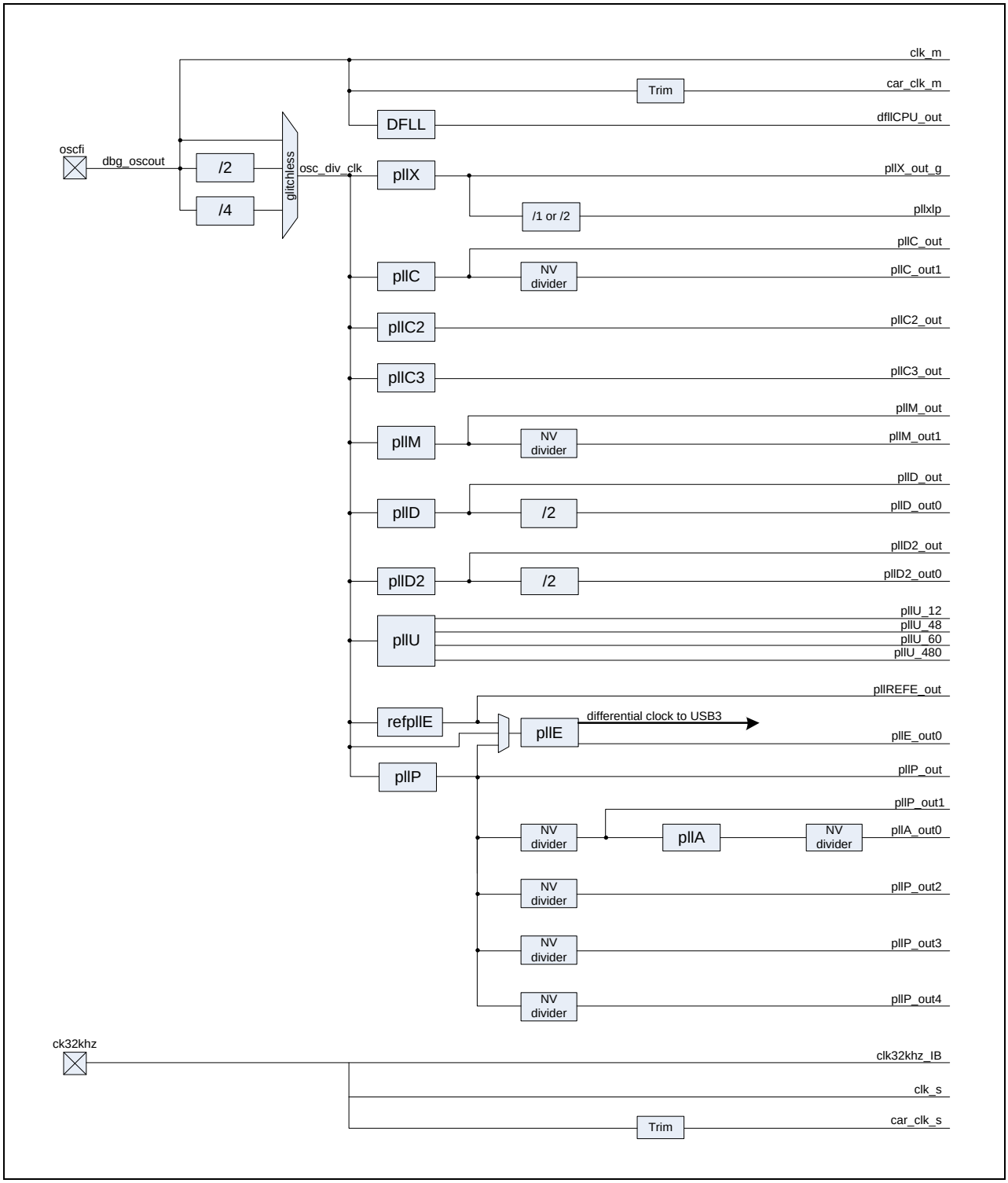
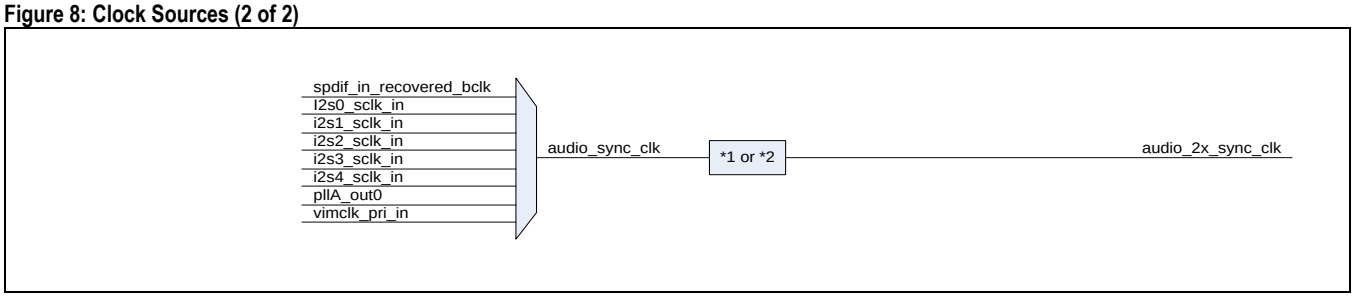


Figure 8: Clock Sources (2 of 2)



The table below gives a quick overview of the root clock generators. It includes the clock sources and the divider type and size. For example, the ACTMON root clock has 6 clock sources (000=pllP_out, 001=pllC2_out, 010=pllC_out, 011=pllC3_out, 100=ck32khz_IB, 110=dbg_oscout) and has an 8-bit wide U7.1 fractional divider. The default reset value is shown as an underlined value such as 6.

Table 15: Root Clock Sources

	audio_2x_sync_clk	ck32khz_IB	dbg_oscout	dfllCPU_out	pllA_out0	pllC_out	pllC_out1	pllC2_out	pllC3_out	pllD_out0	pllD2_out0	pllE_out0	pllM_out	pllM_out1	pllP_out	pllP_out2	pllP_out3	pllP_out4	pllREFE_out	PllU_48	pllU_60	pllU_480	pllX_out	Have super clock divider ? (# of bits)	Have U7.1 divider? (# of bits)
ACTMON		4	***6***			2		1	3						0										8
ADX0			***6***		0	2		1	3						4										8
AMX0			***6***		0	2		1	3						4										8
AUDIO	7		***6***		0	2		1	3						4										8
CILAB			***3***			1									0										8
CILCD			***3***			1									0										8
CILE			***3***			1									0										8
CPU		2	***0***	9[c] 15		1		6	7				3		4			5					8[c] 14	8[d]	8
CPULP		2	***0***			1		6	7				3		4			5					8[a,c] 14[a]	8[d]	8
CSITE			***6***			2		1	3				4		0										8
DAM[0:2]	7		***6***		0	2		1	3						2										8
DISP1, DISP2			***6***		3	4				2	5		1		0										8
DSIA_LP			***3***			1									0										8
DSIB_LP			***3***			1									0										8
DVFS_REF			***6***			2		1	3				4		0										8
DVFS_SOC			***6***			2		1	3				4		0										8
EMC			***3***			2		5	6				0, 4[b]		1										8
ENTROPY			***1***												0										8
EPP					6	2		1	3				***0***		4										8
EXTPERIPH [1:3]		1	***3***		0							4			2										8
GR2D					3	1		4	5				***0***		2										8
GR3D					6	2		1	3				***0***		4										8
HDA			***6***			2		1	3				4		0										8
HDA2CODEC_2X			***6***			2		1	3				4		0										8
HDMI			***6***		3	4				2	5		1		0										8
HDMI_AUDIO															***0***									-	-
HOST1X					6	2		1	3				***0***		4										8
HSI			***6***			2		1	3				4		0										8
I2C[1:5]			***6***			2		1	3				4		0										16
I2C_SLOW		4	***6***			2		1	3						0										8
I2S[0:4]	1		***3***		0										2										8
LA			***6***			2		1	3				4		0										8
LFSR													***0***											-	-
MIPI_CAL																	***0***							-	-
MIPI_IOBIST																	***0***							-	-
MSELECT			***6***			2		1	3				4		0										8
MSENC					6	2		1	3				***0***		4										8
NANDFST			***6***			2		1	3				4		0										8
NAND_ SPEED			***6***			2		1	3				4		0										8

	audio_2x_sync_clk	ck32khz_IB	dbg_oscout	dfllCPU_out	pllA_out0	pllC_out	pllC_out1	pllC2_out	pllC3_out	pllD_out0	pllD2_out0	pllE_out0	pllM_out	pllM_out1	pllP_out	pllP_out2	pllP_out3	pllP_out4	pllREFE_out	PllU_48	pllU_60	pllU_480	pllX_out	Have super clock divider ? (# of bits)	Have U7.1 divider? (# of bits)
NOR			***6***			2		1	3				4		0										8
OWR			***6***			2		1	3				4		0										8
PWM		4	***6***			2		1	3						0										8
SBC[1:6]			***6***			2		1	3				4		0										8
SDMMC[1:4]			***6***			2		1	3				4		0										8
SE			***6***			2		1	3				4		0										8
SOC_THERM					6	2		1	3				0		4										8
SPDIF_OUT	1		***3***		0										2										8
SPDIF_IN						2		1	3				4		***0***										8
SYS (sclk,hclk,pclk)		6	***0***			5	1							7	3	4		2						8	
TRACECLKIN			***6***			2		1	3				4		0										8
TSEC			6			2		1	3				***4***		0										8
TSENSOR		***6***	4			2		1	3						0										8
UART[A:D]			***6***			2		1	3				4		0										17
UART_FST[A:D]																	***0***							-	-
VDE			***6***			2		1	3				4		0										8
VFIR			***6***			2		1	3				4		0										8
VI					6	2		1	3				***0***		4										8
VI_SENSOR					6	2		1	3				***0***		4										8
XUSB_CORE						1		3	4						***0***				2						8
XUSB_ FALCON						1		3	4						***0***				2						8
XUSB_FS															1					2		***0***			8
XUSB_HS																					1	***0***			8
XUSB_SS						2		4	5						1				3		6	***0***			8

Note:

a. For clock source pllX_out, this is an output of a mux that can select between the pllX_out clock and a predivide-by-2 pllX_out clock. For clock source dfllCPU_out, this is an output of a mux that can select between the dfllCPU_out clock and a predivide-by-2 dfllCPU_out.

b. When EMC clock source = 4, the U7.1 divider will be bypassed/ignored. This setting will give a very short low jitter clock path from pllM_out to EMC clock.

c. When clock source = 8 or 9, in addition to being selected into the glitch-less switching logic in CAR, the selected clock will also bypass the U7.1 divider creating a short low jitter clock path to the output clock.

d. The Clock Skipper is controlled by SOC Therm hardware.

6.2.3 Generic Clock Core Structure

Root clock generation is typically a clock source selection mux followed by a U7.1 fractional divider. The clock source selection mux is made glitch-less by the addition of a state machine that gates off the clocks that might be impacted by the glitch, switching the clock source, and then ungating the clock.

Changing the fractional divider ratio is also glitch-less and will use one of two mechanisms. When used in a switch with a clock source selection mux, the switching state-machine described above is used. When used without a clock source selection mux such as in the pllP_out1,2,3,4 dividers, the divider is not stopped and seamlessly switches to the new frequency at the end of a clock period. In this latter scheme, no dead cycles exist– between the original frequency and then the newly requested frequency.

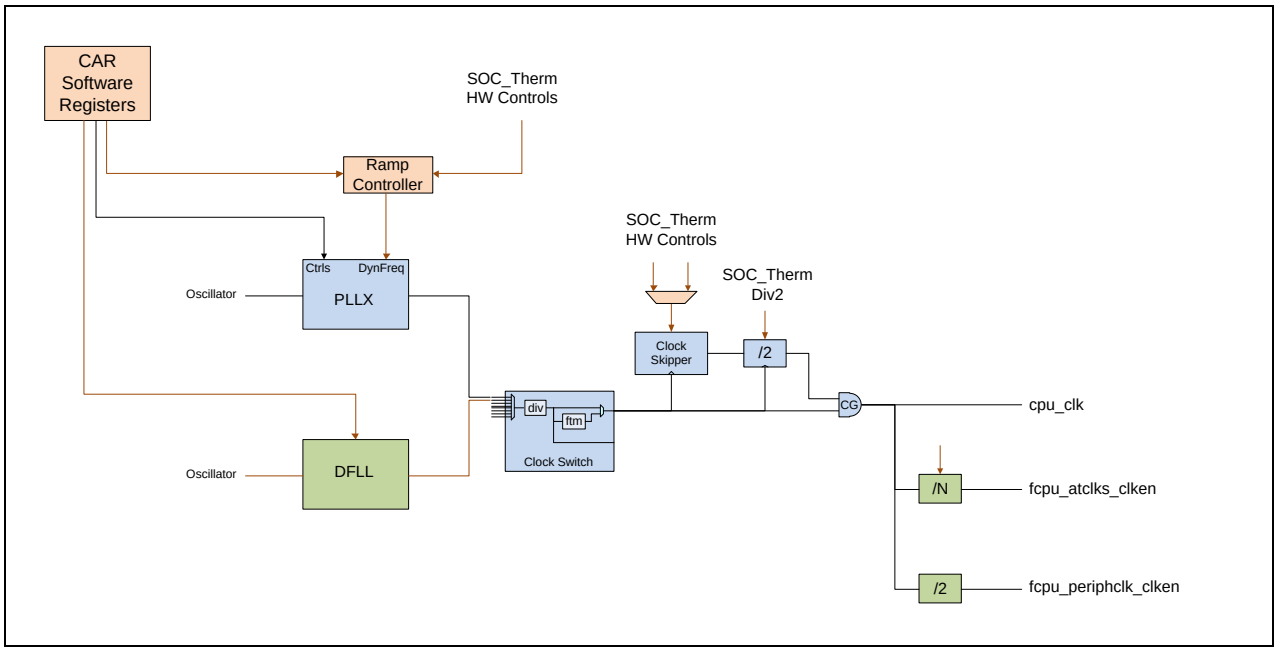
In all structures above, when the clock is gated off, it occurs at the end of the clock period so that 1) there are no runt pulses, and 2) the output clock will be stopped low.

6.2.4 Custom Core Clocks

6.2.4.1 FCPU Clock

The following figure shows a block diagram of the components of the Fast CPU (FCPU) clocking.

Figure 9: FCPU Clock Generation



DFLL

The primary clock source for the FCPU is the DFLL. The DFLL is based around a ring oscillator with supporting structures for di/dt management and frequency control.

PLLX

PLLX is one of two possible primary clock sources for the FCPU. It contains two dynamic frequency mechanisms: a fast one that could overshoot and a slower one with no overshoot.

Clock Skipper

An M/N clock skipper that is managed by the SOC Thermal module.

SOC Therm Div 2

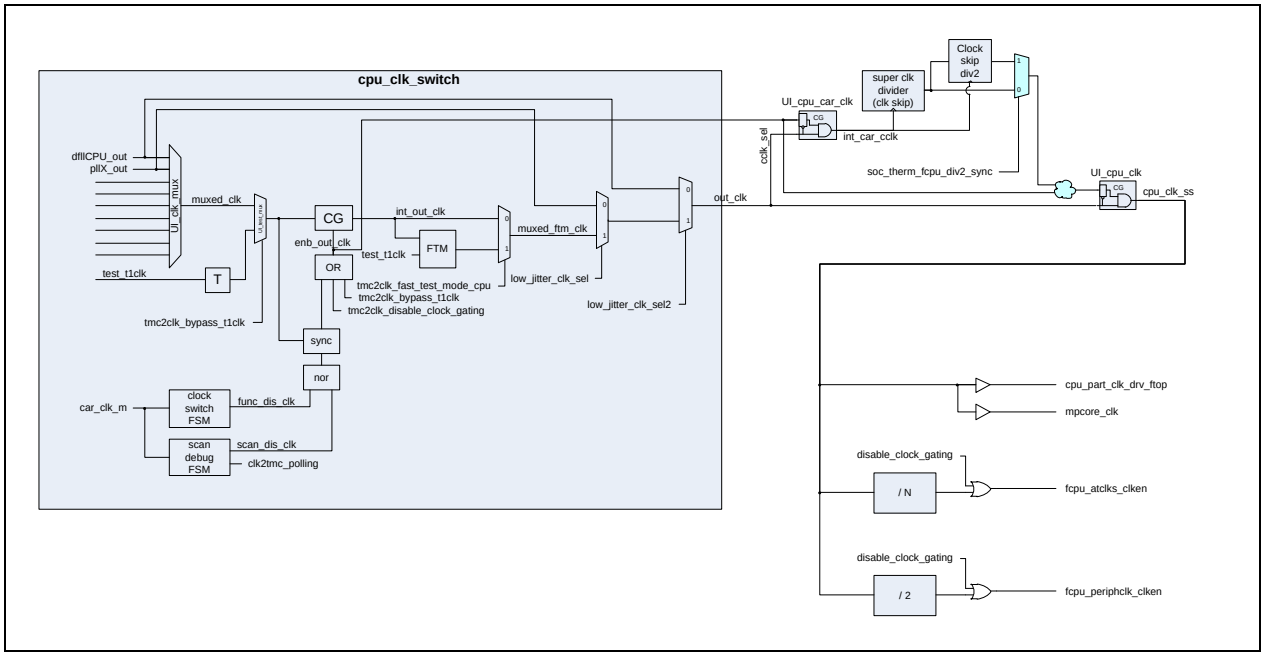
This logic will slow the CPU clock to manage thermal runaway.

Clock Switch

The clock switch consists of a clock source selection mux that can switch glitch-lessly between a number of clock sources including the PLLX and DFLL. This is followed by a programmable U7.1 divider. For low jitter when the divider is not required, a direct bypass path from PLLX or DFLL inputs to the output is also provided.

The following figure shows the details of the clock switch, clock skipper, SOC Therm Div 2, and root clock generation. The clock sources to the clock source selection mux are described in the table entitled “Root Clock Clock Sources” in the “Main Clock Sources” section.

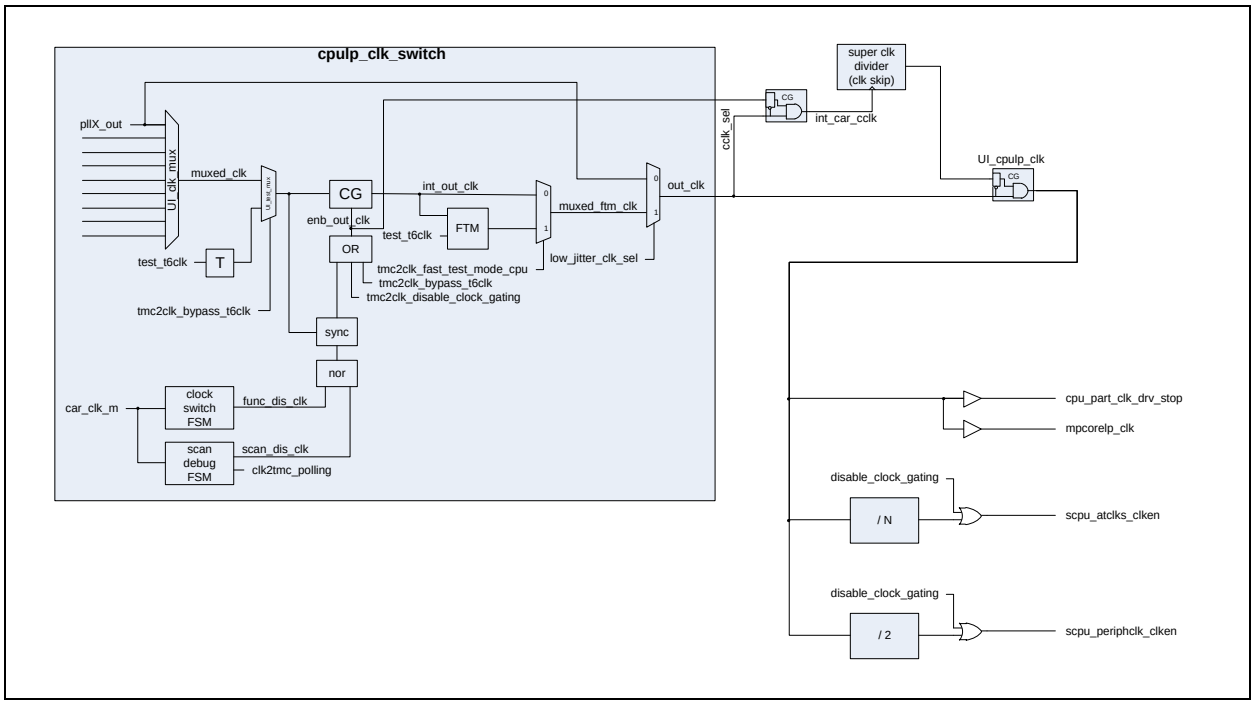
Figure 10: FCPU Clock Details



6.2.4.2 SCPU Clock

The following figure shows the details of the Clock Switch, Clock Skipper, and root clock generation for the Slow CPU (SCPU). The clock sources to the clock source selection mux are described in the table entitled “Root Clock Clock Sources” in the “Main Clock Sources” section.

Figure 11: SCPU Clock Details



6.2.4.3 GR3D Clock Switch

The hardware-controllable M/N clock skipper on the GR3D clock is managed by the SOC Therm module hardware. The clock sources to the clock source selection mux are described in the table entitled “Root Clock Clock Sources” in the “Main Clock Sources” section.

6.2.4.4 EMC/MC Clock

The PLLM’s primary purpose is to clock the MC and EMC. The EMC/MC clocking structure is complicated due to the need to support high speed DRAMs, avoid clock switching when DRAMs are being accessed, and conserve power.

When changing the EMC frequency, internal logic sequencing handles the requirement that the DRAMs must be placed into self-refresh before the clock frequency is changed. Certain register fields when written do not get immediately applied but rather are applied at the correct time during the sequencing. These registers are referred to as "shadowed". Other register fields initiate the sequencing state-machine and therefore should be written last. At the end of the sequencing, the DRAMs are restored to being operational.

The reprogramming sequence is:

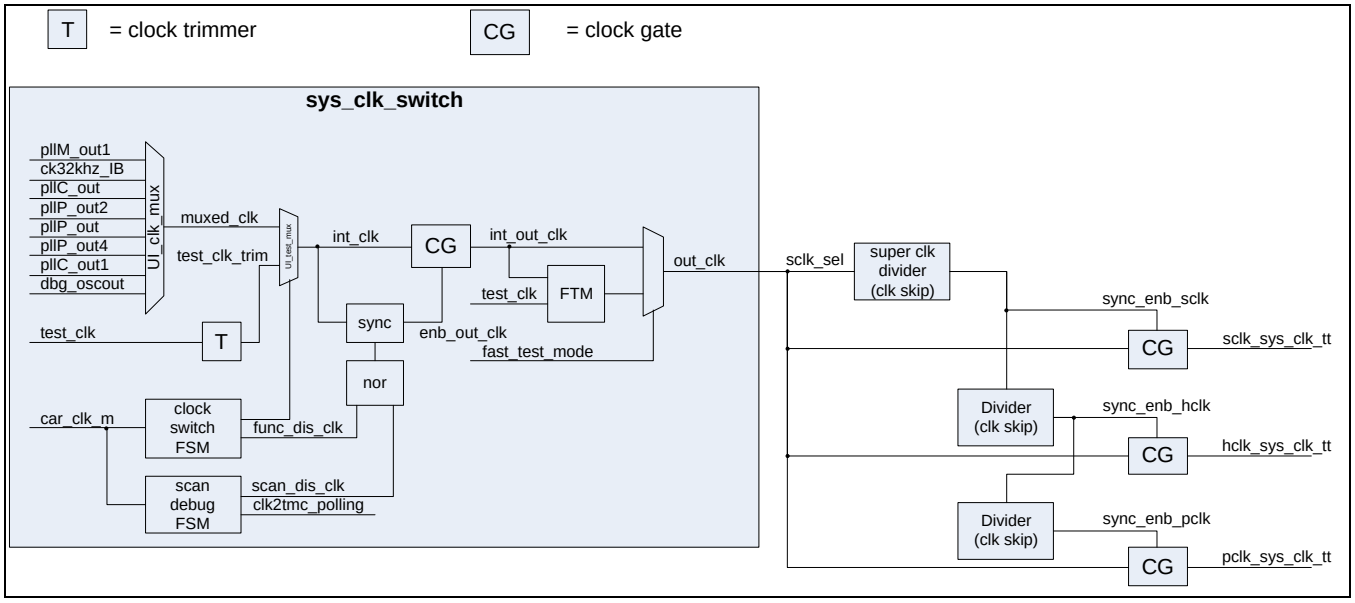
1. Program the MC/EMC shadow registers corresponding to the new frequency. These registers are “shadowed”, as noted in their descriptions, and will not impact the hardware until later during the clock change sequence.
2. Program the EMC clock change FIFO (CCFIFO) with the pre-/post-clock change sequence.
3. Program CAR CLK_SOURCE_EMC register to trigger a clock change. At this point hardware takes the following actions:
   - The CAR block asserts the clock change request to the EMC.
   - The EMC stops MC transactions, flushes its internal outstanding requests, executes MRSs/enter self-refresh, and then updates the timing to copy the EMC shadow
   - Register contents (including CDB phase select) to EMC current register.
   - The EMC asserts clock change acknowledge to the CAR block.
   - The CAR stops the clock to PLLM/CDB, updates to the new clock source, and passes the EMC CDB phase select control to the CDB.
   - The CAR re-enables the clock to PLLM/CDB.
   - The CAR deasserts the clock change request to the EMC (telling it the clock change is done).
   - The EMC unblocks MC transactions, deasserts clock change acknowledge.
   - The EMC executes the post clock change sequence (self-refresh exit, MRSs).
   - Normal memory access resumes.

Software monitors the clock change complete interrupt status from the EMC register to know when the clock change is done.

6.2.4.5 SYS Clock Generation (SCLK, HCLK, PCLK)

The system clock sclk is used by the AVP sub-system (COP), hclk is the AHB bus clock, and pclk is the APB bus clock.

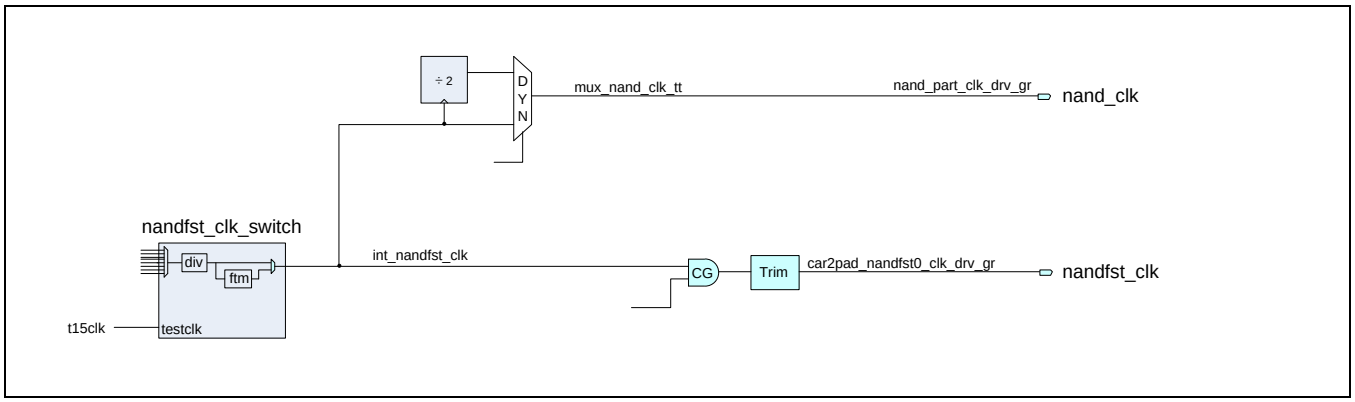
Figure 12: SCLK Clock Generation



6.2.4.6 NAND/NANDFST Clock

The NANDFST clock can be either the same or twice the frequency of the NAND clock as shown in the figure below.

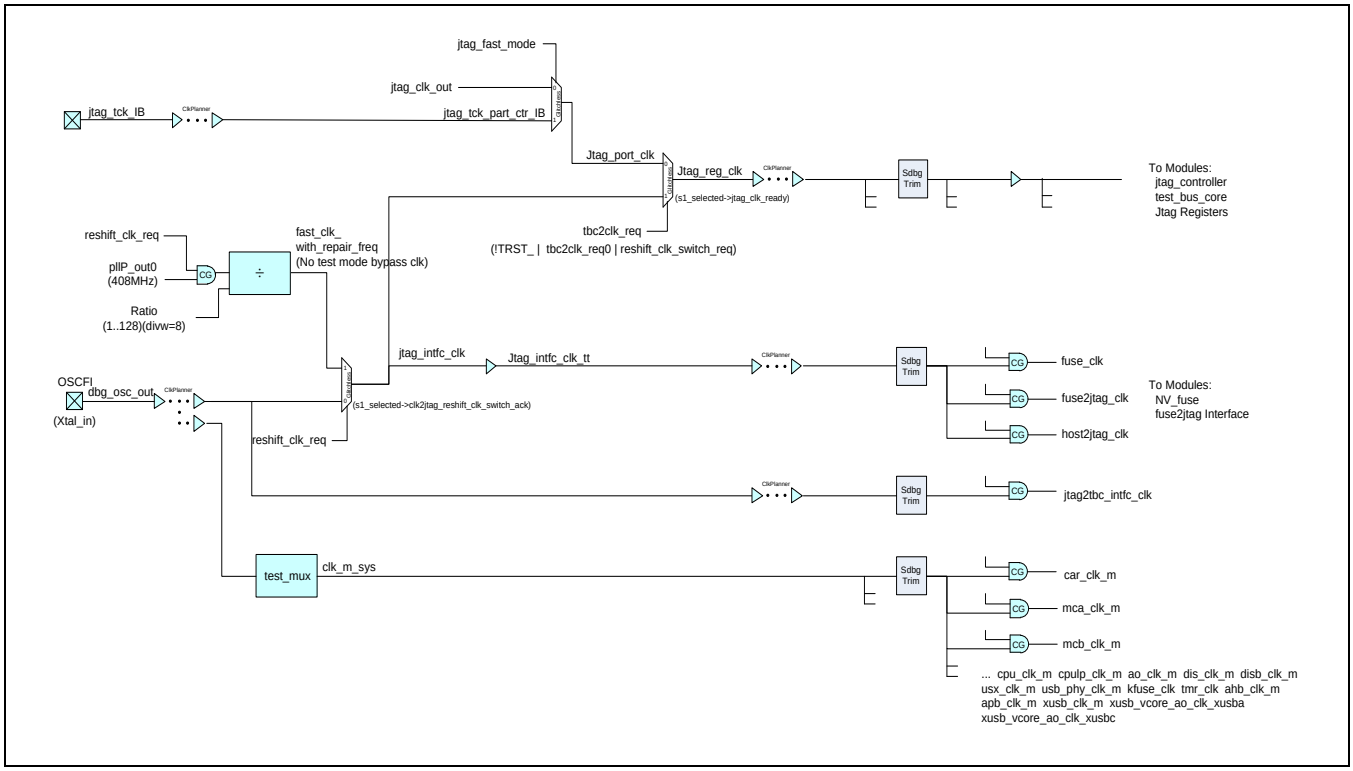
Figure 13: NAND/NANDFST Clock Generation



6.2.4.7 Jtag_ref_clk and fuse_clk

The RAMs in the Fast CPU need to support RAM repair. Since the power rail to the FCPU will be shut down in modes such as LP1, the RAM repair information will be lost. When power is returned, the RAM repair information needs to be reprogrammed. A faster clock is needed to avoid delays in bringing the FCPU online. Additionally, a synchronous clock relationship is temporarily needed between the fuse and the re-repair logic running on the JTAG clock.

Figure 14: Jtag_ref_clk and fuse_clk Clock Generation



6.3 PLLs

6.3.1 DFLLCPU

DFLLCPU is a dedicated clock source for the Fast CPU. The DFLL is based on a ring oscillator and translates voltage changes into frequency compensation changes needed to prevent the CPU from failing.

6.3.2 PLLX

Tegra 4 devices require an alternative clock source in addition to DFLLCPU. Additionally, for EDP management and to avoid creating di/dt problems, this PLL needs a dynamic frequency changing mechanism.

Similar to prior Mobile products, PLLX is dedicated for this purpose. This PLL will feed to both the Fast CPU Cluster and Shadow CPU. This PLL cannot be easily used opportunistically for other units.

6.3.3 refPLLE, PLLE, USB3 Brick PLLs

USB 3.0 requires a spread spectrum (SS) clock. It also has stringent jitter requirements to be met. A spread-spectrum PLL with high VCO frequency (for low jitter) meets these requirements. A dedicated PLL (PLLE) is used for this purpose.

In addition to PLLE, to ensure that oscillator clock jitter gets filtered out prior to feeding it to PLLE, a reference PLL (refPLLE) is required. There are additional PLL(s) in the pad brick used for physical layer signaling for USB 3.0. For cases where spread spectrum might not be required and jitter may not be an issue, an option is kept to bypass PLLE to save power. This option requires that the crystal be 12 MHz or 48 MHz to be able to create the required PLLE output frequency of 100 MHz.

6.3.4 PLLU and UTMI PLL

USB 2.0 mode requires 8 phase data sampling logic per USB 2.0 port to capture data. This requires a dedicated PLL (UTMI_PLL) with 2 data sampling logic ports. The dedicated PLL (PLLU) is used to generate the low jitter reference clock for UTMI_PLL. There is a provision to bypass PLLU for cases where a slightly higher jitter is acceptable for UTMI PLL reference clock.

6.3.5 PLLM

The DDR interface in Tegra 4 devices is required to operate at 1600 MT/s. This means that the EMC should work at 800 MHz and the MC should work at 400 MHz. The memory subsystem (DDR, EMC, and MC) is required to support a number of discrete frequencies like DDR 1600, DDR 1333, DDR 1066, DDR 800 as well as slower frequencies to conserve power. A dedicated PLL (PLLM) is used for memory clocks.

DDR interface signaling also needs a DLL to support ¼ clock shift to reliably capture/drive DQ. In addition, a CDB (clock distribution buffer) is required to get a low jitter/skew clock for each byte/command group.

6.3.6 PLLA and PLLP

All audio clocks are a multiple of the sampling frequency which can be 32 KHz, 44.1 KHz or 48 KHz. A frequency of 56.448 MHz or 73.728 MHz can be used to generate any of these sampling rates. It's not possible to generate both of these frequencies from the osc_freq by just using PLLA so a cascade of 2 PLLs, PLLP to PLLA, is required to generate the audio clocks.

PLLP is fixed at generating 408 MHz, and PLLA takes in 9.6 MHz (PLLP / 42.5) to generate the two required audio frequencies.

Because PLLP generates a fixed frequency clock for audio, it can also be divided down and used for other modules which might need a fixed frequency clock.

Most audio codecs have their own PLL, which turns out to be more power efficient, so it is possible that PLLA does not get used in a typical system for audio. In such a case PLLA can be used as a general purpose PLL.

6.3.7 Display Clocks and PLLs (PLLD and PLLD2)

Tegra 4 displays support the same 2 heads as in Tegra 3 displays. Following are the different display clocking requirements:

- Each display head requires an independent pixel clock which in worst case would need a dedicated PLL to generate the required frequency as there are too many precise clock frequencies that are required for pixel clock.
- DSI/HDMI interfaces require a differential clock to be sent out. So a dedicated PLL that can generate differential clocks is required for each one. In addition, DSI requires some PHY specific functionality to be present in the PLL.

PLLD and PLLD2 are two dedicated PLLs (with differential clock outputs and some DSI specific functionality) which should be used for DSI0 and DSI1. These PLLs can be configured to generate the pixel clock from any oscillator frequency. These PLLs consume high power (estimated at 4 times a regular general purpose PLL) and should be used only if DSI is required. For HDMI, the pad brick has PLLs which are used for interface signaling. These PLLs expect a pixel clock as an input.

6.3.8 PLLC2 and PLLC3

The A7-AVP (300MHz), MSENC (333 MHz), imaging (VI/ISP – 300 MHz), VDE (366 MHz) and various other high frequency peripherals have different maximum frequency requirements. PLLC2 and PLLC3, can go up to 600 MHz, are used for these modules.

6.3.9 PLLC

PLLC is a General-Purpose PLL. PLLC has dynamic frequency switching support, 300 MHz to 700 MHz, without overshoot. PLLC is available to most of the same modules as PLLC2 and PLLC3.

6.4 Reset Architecture

The external PMC provides the primary power-on reset (SYS_RESET_N_). Apart from POR, some other events can also result in a system reset. These are:

1. Reset due to an indication from a thermal sensor. The thermal sensor module would assert a reset signal tsensor2pmc_reset, which would in turn generate a reset for the whole chip and in the process deassert tsensor2pmc_reset itself.
2. Expiry of watchdog timer. The watchdog timer has a counter which is loaded with an initial value and ticks on periodic intervals. Once the counter expires, it reloads itself with the initial value, increments an expiry count, and generates an event for software. If software acknowledges the event, the expiry count is cleared and the counter is loaded back with initial value. This process continues indefinitely. Depending on the current expiry count, the event generated for software could be different. There are 2 types of watchdog timers in Tegra 4 systems depending on how they generate the reset:
   - Deadman timer (referred as WDT in reset code): This is the legacy implementation in which:

     1st expiry an interrupt is issued

     2nd expiry a reset is issued. This reset, however, resets only a subset of units.
   - Watchdog timer (referred as WDT2 in reset code):

     1st expiry – interrupt is issued

     2nd expiry – FIQ is issued

     3rd expiry – CPU reset is issued

     4th expiry – full system reset – This is the relevant reset for this section
3. Software reset. This reset is controlled by a configuration bit in the PMC address space (main_swrst). Once asserted, this result in a reset generation for the whole chip and in the process deasserts itself.
4. LP0 wakeup reset. This is controlled by the logic within the PMC.

The POR (SYS_RESET_N_) is deasserted by the external PMC after the power sequencing is done and after the RTC clock (clk_32KHz) and crystal clock is already running. For the other sources, the reset is originated within the Tegra 4 hardware.

6.5 Power Gating and Ungating

6.5.1 Clamp/Reset/Clock/PG-Enable Sequencing

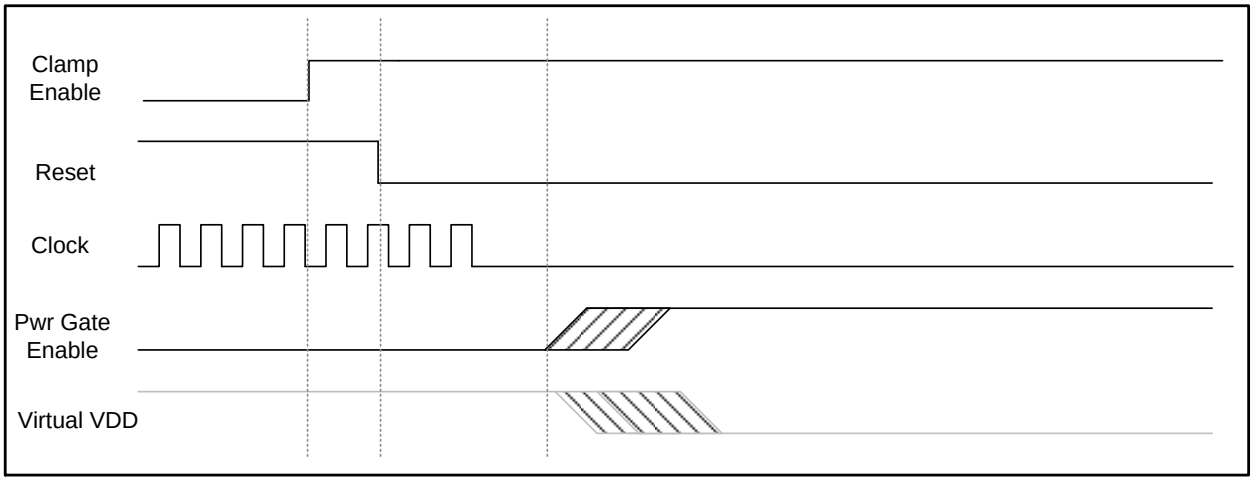
6.5.1.1 Power Gating

For power-gating (powering off) a partition, the clamp, reset, clocks, and PG-enable (also known as sleep-enable) controls must be sequenced as shown in following diagram. For CCPLEX PG partitions, this sequencing is ensured by hardware (when power-gating is done via flow-controller). For SoC (non-CCPLEX) PG partitions, this sequencing needs to be ensured by software.

In general, clamp-enable should be asserted before reset (as shown in the diagram). This works with synchronous or asynchronous clamping. However, if a unit can guarantee that its output signals will have the same clamp values as their pre-clamp values (idle values) as well as their reset values, then clamp and reset ordering is not critical. Note it is non-trivial to verify the above guarantee.

Also, it is recommended to have synchronous clamps. However, if a unit can guarantee that its output signals will have the same clamp value as their pre-clamp values (idle values) as well as their reset values, then asynchronous clamping can be used. It is non-trivial to verify the above guarantee.

Figure 15: Power-Gating Timing Sequence for Clamp/Reset/Clock/PG-Enable

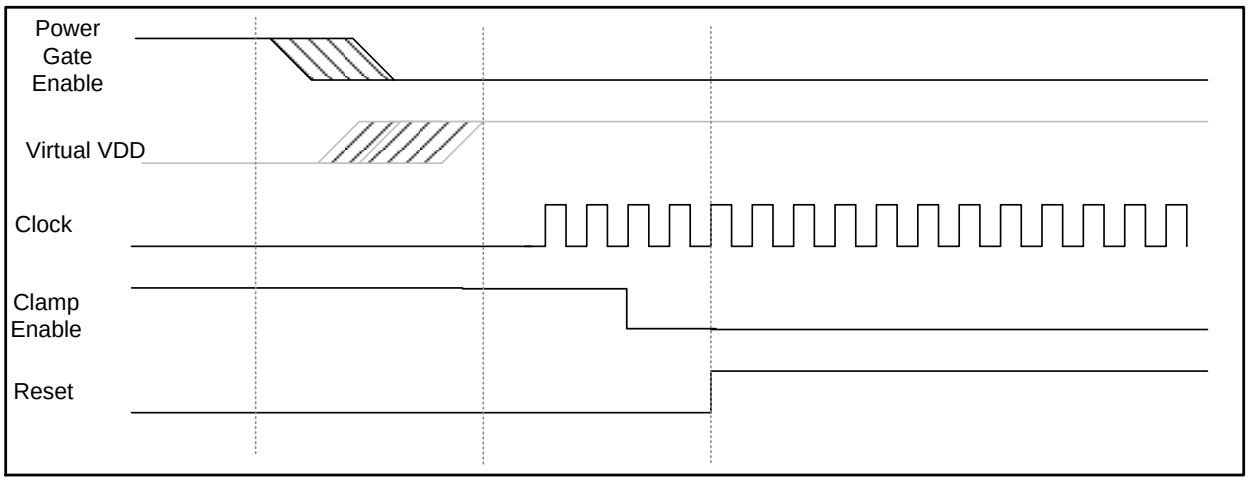


6.5.1.2 Power Ungating

For power-ungating (powering on) a partition, we have to ensure that clamp, reset, clocks, and PG-enable (sleep-enable) controls are sequenced as shown in the following diagram. For CCPLEX PG partitions, this sequencing is ensured by hardware (when power-gating is done via the flow controller). For SoC (non-CCPLEX) PG partitions, this sequencing needs to be ensured by software.

In general, the clamp enable should be deasserted before reset (as shown in the diagram). This works with synchronous or asynchronous clamping. However, if a unit can guarantee that its output signals will have same clamp values as its post-clamp values (idle values), then clamp and reset ordering is not critical for power ungating. Note it is non-trivial to verify the above guarantee.

Figure 16: Power-Ungating Timing Sequence for Clamp/Reset/Clock/PG-Enable



6.5.2 Power-Gating Controller

The PMC provides power-gating controllers as programmable sequencers. When the power gating/ungating in is triggered, the controller sequences PG-enable signals with a programmable delay (measured in APB clock cycles) between consecutive “zones” PG-enable controls. There are two types of power-gating controllers: CPU and SOC power-gating controllers. The following sections describe both types of controllers.

6.5.2.1 SOC Power-Gating Controller

The SOC power-gating controller is used to sequence power gating of SOC (including shadow CPU cluster) power partitions. It uses 8 zones and shares the same programming register for inter-zone delays. The zones are powered up in one order and powered down in reverse order, using the same inter-zone timings. So only one set of delays are used for both powering down and up.

For all non-CPU SOC partitions, power-gating can only be turned on/off by direct register write. For CPU (shadow) SOC partitions, power-gating can be turned on/off via flow-controller. Refer to “SOC Power Gating Controllers” in the PMC section.

GPU Power Gating

GPU power gating is an exception to power-gating of other SoC partitions. In Tegra 4 devices, the GPU (also known as TD) will have 6 partitions, which should be seen as one ELPG (Engine Level PG) by software. They should be powered on/off sequentially due to power restrictions. The PMC would sequence each of the TD partitions using the SOC power-gating controller, and would have a programmable inter-partition delay. The operation would be transparent for software. The PMC would also have an option of sequencing all 6 partitions in parallel to shorten power-gating latency.

Each of the GPU partitions would have its separate reset (from the CAR) and clamp (from the PMC). All of the GPU partitions can be clamped at the same time (and similarly reset at the same time). In other words, software should be able to treat the GPU as one engine for sequencing clamp/reset/clock/power-enable as described in previous sections.

6.5.2.2 Fast CPU Power-Gating Controller

The fast CPU power-gating controller is used to sequence power-gating of fast CPU partitions. All CPU partitions (CE0/1/2/3 and C0NC) share the same programming register for inter-zone delays, which are different from inter-zone delays in SoC partitions. The zones are powered up in one order and powered down in reverse order, using the same inter-zone timings. So only one set of delays are used for both powering down and up. For fast CPU partitions, power gating can be turned on/off via the flow controller. Refer to “CPU-G Power Gating Controllers” in the PMC section.

6.6 SRAM Power Gating

Tegra 4 devices contain two distinct types of RAMs: Custom RAMs (FCMs) in the VDD_CPU power domain and compiled RAMs in the VDD_SOC and VDD_AO domains.

6.6.1 Custom RAMs (FCMs) in CCPLEX

All custom RAMs (FCMs) can be power-gated. The following table lists the FCMs in the Tegra 4 VDD_CPU domain (all FCMs are in fast CCPLEX only).

SRAM	Partition	Tiles/ Inst	# Inst	# Cores
eag_l2_data_ram	FL2	4	16	1
eag_l2_tag_ram	FL2	1	32	1
eag_l2_tlb_ram	FL2	1	4	4
eag_l2_dirty_ram	FL2	4	1	4
eag_l2_psq_ram	FL2	1	1	4
eag_l2_snp_tag_ram	FL2	1	4	4
eag_ls_data_ram	FCPU	2	4	4
eag_ls_tag_ram	FCPU	2	4	4
eag_if_data_ram	FCPU	1	16	4

SRAM	Partition	Tiles/ Inst	# Inst	# Cores
eag_if_tag_ram	FCPU	1	2	4
eag_if_btb_ram	FCPU	1	4	4
eag_if_ghb_ram	FCPU	1	6	4
eag_if_ip_ram	FCPU	1	1	4

6.6.2 RAMs in SOC

The RAMs in the following table support power gating.

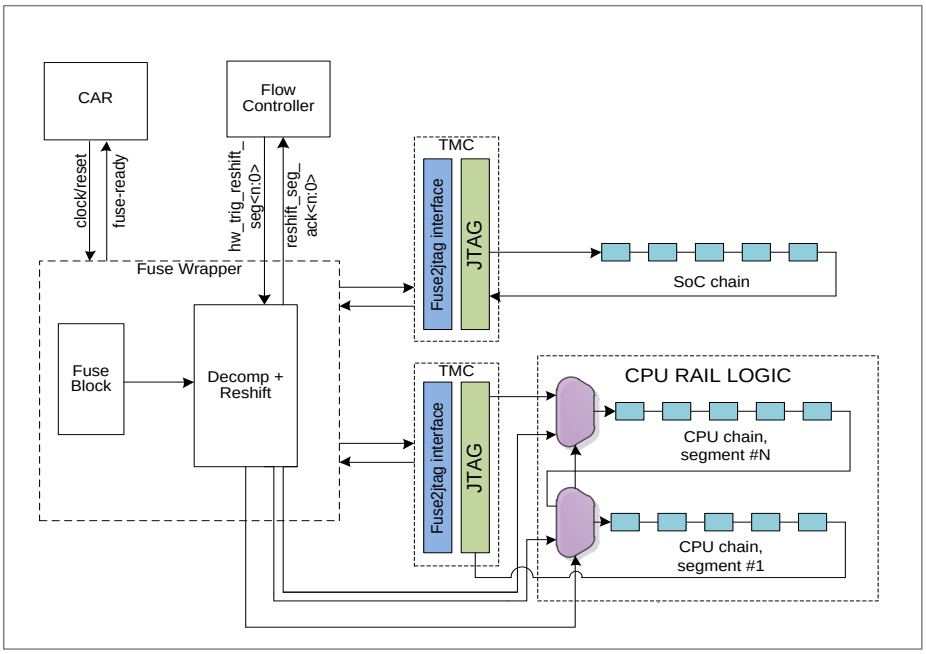
RAM	Partition	Count
RAMSP_2048X32_ABE8GR_M8_B1	SL2 (Companion core L2 Data)	64
RAMSP_2048X32_ABGPR_M8_B1	MSENC (TEB)	28
RAMSP_2048x541_BGPR_M4_B1	DISP	1
RAMSP_832x541_BGPR_M4_B2	DISP	1
RAMSP_768x541_BGPR_M4_B3	DISP	1

6.7 SRAM Repair and Re-repair

To repair faulty SRAMs, a set of repair flops are used in SRAMs to hold repair mux controls. At initial power up, these flops are programmed by fuse-value shifted into “repair chains”.

Tegra 4 devices use PAGO cells for repair flops, which hold repair information when an SRAM is power-gated. However, when the corresponding rail is powered off, PGAO repair flops are powered off.

Figure 17: SRAM Repair and Re-repair Block View



6.7.1 Number of Repair Chains and Segments

Tegra 4 devices have two repair chains:

- Repair chain for SoC-rail RAMs (including RAMs in shadow CPU cluster), referred to as the SoC chain
- Repair chain for CPU-rail RAMs, referred to as the CPU chain

To speed up the CPU RAMs repair, the CPU repair chain is segmented into 8 segments. The segments of a repair chain can be shifted-in one (or more) segments at a time.

6.7.2 Repair

At SoC rail power-up, the CAR block deasserts reset to the fuse, repair, and reshift logic (referred to as fuse wrapper) while holding reset for other SoC and CPU blocks (especially for boot blocks). At the deassertion of reset, the fuse wrapper reads fuses and shifts repair values into SoC and CPU repair chains, one chain at a time.

At this stage in the boot, the SoC rail is powered on, but the CPU rail is powered off. So repair to the CPU chain fails (at boot time).

The SoC repair chain needs to be repaired only at boot (cold or warm, also known as LP0). But the CPU rail is powered off, so the CPU repair chain needs to be re-repaired every time the CPU rail is powered up.

6.7.3 Re-repair

At boot-time, repair fails because the CPU rail is powered off. However, the “reshift” block snoops the fuse values (being shifted into the CPU chain), and stores them in an SRAM. Subsequently, the CPU chain is re-repaired by triggering the reshift block, which uses repair values from this SRAM.

The CPU chain is re-repaired every time the CPU rail is powered up (including the first time during boot). The following steps describe the sequence of CPU rail powering up:

- CPU rail is powered up
- CPU repair chain is re-repaired
- CPU reset and clamps are deasserted

When the CPU rail is powered up the first time (during boot), software triggers the CPU chain re-repair by programming the flow-controller register. Subsequently, flow-controller hardware triggers CPU chain re-repair. In either case, the flow controller requests the “reshift” block to perform re-repair.

6.7.3.1 Re-repair Programming Guide

Re-repair is required every time the CPU rail is powered up. However, re-repair needs to be explicitly triggered by software during cold or warm (LP0 exit) boot as described in the following sequence. The re-repair required during cluster-switch (from cluster1 to cluster0) or LP2 exit with CPU rail power-up is automatically done by hardware.

The re-repair frequency can be programmed (upfront) by writing to the CAR CLK_RST_CONTROLLER_PLLP_RESHIFT register. Software programs this value at cold/warm boot, and it does not change after that.

If need be, software can configure the re-repair frequency to be different for cluster-switch and/or LP2-exit by programming the appropriate value in the CAR CLK_RST_CONTROLLER_PLLP_RESHIFT register before cluster-switch or before going into LP2. As a result, when hardware triggers re-repair, it would use the frequency configured earlier by software.

Software CPU-Rail-Power-Up Sequence including RAM Re-repair

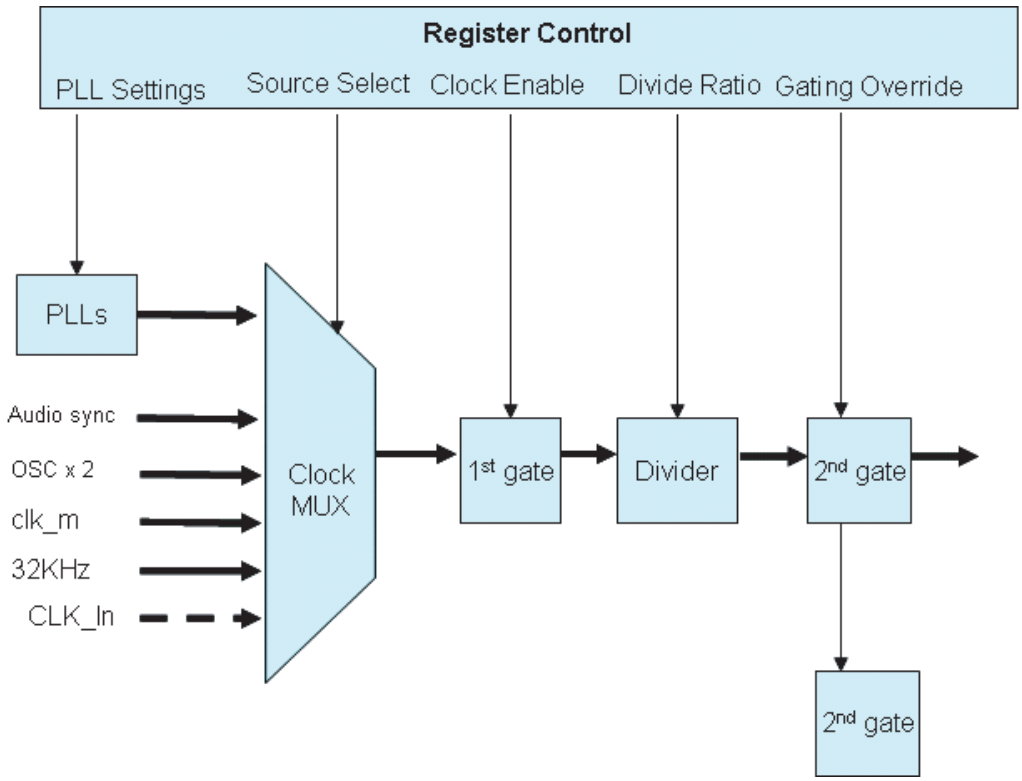
- Power-up CPU rail by direct register write or via PMIC I2C
- Wait for CPU rail power up (by reading PMC_PWRGATE_STATUS[CARIL] status)
- Trigger RAM repair (by writing FLOW_CTLR_RAM_REPAIR[REQ])
- Wait for RAM repair to be done (by reading FLOW_CTLR_RAM_REPAIR[STS])
- Continue the rest of power-up sequence as without re-repair, that is, clamp removal and reset deassertion).

6.8 Software Features and Programming Model

6.8.1 Clock Control

The figure below illustrates the software clock control model.

Figure 18: Clock Control



6.8.2 PLL Programming

PLL output frequencies are programmed by setting their N, M, and P values. The governing equations are:

VCO = (Fi / M) * N,

Fo = VCO / (2 ^ P)

where *Fo* is the output frequency from the PLL.

> **Note:** Not all PLLs have the simple mapping for the P value listed above. Contact your NVIDIA representative for assistance.

There are three requirements for each PLL that must be complied with:

- Each PLL has a legal input frequency (*Fi*) range.
- Each PLL has a legal comparison frequency (*CF*) range, where *CF = Fi / M*
- Each PLL has a legal *VCO* frequency range, where *VCO = CF * N*.

To change a PLL, do the following:

- Ensure that no enabled module is using the PLL that will change.
- Program the new PLL settings
- Wait for PLL stabilization
- Change the divider values for each clock that will use the PLL to divide-down to the target frequency, and change the module clock sources for all modules that will use the new PLL settings.

6.8.3 Clock Division Control

The ratio as defined by the *1/divisor* for an 8-bit fractional divider, of 7 bits of *d* and 1 bit of *h* is:

divisor = (ddddddd + 1) + (h * 0.5)

The divisor for UART 16b integer divider it is:

divisor = (dddddddddddddddd)

The divisor for the I2C 16b integer divider it is:

divisor = (ddddddddddddddd + 1)

6.8.4 Changing Clock Sources and Clock Dividers

Changing a clock source (except the clock sources to the audio_sync_clk) to a running module is "glitch free". The clock source and divide ratio can be set concurrently without creating spurious frequencies. For the audio_sync_clk, the clock source select needs to be set up before changing the device clock source to use that audio clock or to enable the device which uses that audio clock.

The glitch-free clock divider to a running module can also be changed. All modules support changing the clock divider ratio without disabling the clock; write the new divider to the appropriate register See the section on Power below.

To change the clock source and divider (either after system reset or for any other reason), follow this sequence:

Sequence 1 to change clock source and divider:

1. Assert reset to the module if it is not already asserted
2. Enable clock to the module if the clock is not already enabled
3. Change the clock divider and/or the clock source to the module
4. Wait 2 µs for the clock to flush through the pipe / logic.
5. Deassert reset to the module.

6.8.5 Clock Gating (Override)

Please refer to the register definition section for more information.

6.8.5.1 Power

The following guidelines should be followed in pursuit of the lowest use-case power consumption:

- Target the least number of PLLs running at their lowest allowable frequencies for the given use-case.
- For each module, use the source with the lowest frequency that provides adequate performance for the use case.
- Turn off all clocks not required for the given use-case - employ maximum clock-gating.
- Use hardware dynamic clock bursting whenever possible. Turn on the desired frequency, burst to completion, and then disable the input frequency (allowing PLLs to be turned off or their output frequencies to be lowered)
- Use the CPU and COP/system super clock divider for lower CPU frequency where possible.
- Disable the oscillator input and/or clock outputs when they are not in use.

6.8.6 Use-Case Restrictions

6.8.6.1 Use of PLLC for Display

Other than PLLD, there is no dedicated PLL for either display controller. PLLD is required when one of the display controllers outputs to DSI, but should be otherwise avoided due to its higher power. When DSI is not one of the output devices, or in dual-display use cases, the additional PLLs PLLC and PLLP must be used. The matrix of choices is shown in the table below.

Table 16: PLL Sources for Display Based on Use-Case

Primary Head	PLL Choices	Secondary Head	PLL Choices
Parallel	P, C, D	--	--
DSI	D	--	--
Parallel	P, C	Sub	P
DSI	D	Sub	P
Parallel	P, C	DSI	D
Parallel	P, C, D	Parallel	P, C, D
Parallel	P, C, D	TVO	P
Parallel	P, C, D	HDMI	P (480p), C, D (720p)
Parallel	P, C, D	CRT	C, D
DSI	D	TVO	P
DSI	D	HDMI	P (480p), C, D (720p)
DSI	D	CRT	C, D

If possible, the first choice should use the fixed PLLP as the Display source clock (divided-down to the required frequency). However, many display resolutions and/or output devices cannot be driven within the required tolerance by a divided-down PLLP.

Second choice is to use either PLLD or PLLC, depending on the acceptable trade-off. If PLLD is used, power consumption is 5x higher than that for PLLC. If PLLC is used, it may compromise other use cases since the Display source PLL must be programmed to a precise frequency to achieve the required Display error tolerance.

Since PLLC is a possible source clock for the AVP, the high-speed serial devices (SFlash and SPI), for the video encoder (MPE), and for 2D and 3D, these devices may not achieve their target (maximum) frequencies when the Display must constrain PLLC directly. An example of this is shown in the table below for a case where external memory is 166 MHz DDR (EMC 2x clock = 333 MHz).

Table 17: Example of Device Frequencies Possible when PLLC Frequency is Constrained for Use by Display

PLLC	PLLP	AVP	HS serial	MPE	2D / 3D
600.0	408.0	150.0	200.0	200.0	216.0
590.0	408.0	147.5	196.7	196.7	216.0
580.0	408.0	145.0	193.3	193.3	216.0
570.0	408.0	144.0	190.0	190.0	216.0
560.0	408.0	144.0	186.7	186.7	216.0
550.0	408.0	144.0	183.3	183.3	275.0
540.0	408.0	144.0	180.0	180.0	270.0
530.0	408.0	144.0	176.7	176.7	265.0
520.0	408.0	144.0	173.3	173.3	260.0
510.0	408.0	144.0	170.0	170.0	255.0
500.0	408.0	144.0	166.7	166.7	250.0

6.8.7 Programming Guide for Power Gating and Ungating

This section covers the software procedures for power gating and ungating of SOC power domains. Power gating and ungating of CPU power domains is not covered here because it is handled through the Flow Controller.

6.8.7.1 Procedure Summary

The general procedure for power gating an SOC power domain is as follows:

1. Write MC register to flush and block MC clients
2. Write CAR register to assert unit resets
3. Write CAR register to disable clocks
4. Write PMC register to gate power domain

The general procedure for power ungating an SOC power domain is as follows:

1. Write PMC register to ungate power domain
2. Write CAR register to enable clocks
3. Write CAR register to deassert resets
4. Write MC register to enable MC clients

In some cases, partitions require specific deviations from the general procedure. Deviations are covered in the domain-specific sections below.

This table summarizes the respective clock and reset bits and MC clients in each SOC power domain.

Table 18: MC Clients for Clocks and Resets per Domain

Domain	Unit Clocks and Resets	MC Clients
VE	VI, CSI, ISP	VI, ISP
TD	3D	NV
VDE	VDE	VDE
MPE	MSENC	MSENC
HEG	2D, EPP	G2, EPP
DIS	DISP1, DSI, DSIB, CSICIL, MIPICAL	DC
DISB	DISP2, HDMI	DCB
XUSBA	XUSB_SS	None
XUSBB	XUSB_DEV	XUSB_DEV
XUSBC	XUSB_HOST	XUSB_HOST

6.8.7.2 Procedures for VE Power Domain

VE Power Gating

1. Flush MC clients VI and ISP by setting the following bits:
   - MC_CLIENT_HOTRESET_CTRL_0.VI_FLUSH_ENABLE
   - MC_CLIENT_HOTRESET_CTRL_0.ISP_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bits are set:
   - VI_HOTRESET_STATUS
   - ISP_HOTRESET_STATUS
3. Set the following bits to assert reset to VI, ISP, and CSI:
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_VI_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_ISP_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_CSI_RST
4. Clear the following bits to disable clocks to VI, ISP, and CSI:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_VI
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_ISP
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_CSI
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = VE

START = ENABLE

6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit VE is set

VE Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   PARTID = VE
   START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the VE bit is cleared
3. Set the following bits to enable clocks to VI and ISP:
   CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_VI
   CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_ISP
4. If the DIS domain is powered up, enable clocks to the CSI by setting the following bit:
   CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_CSI
5. Remove power-gating clamps by writing a 1 to the following bit:
   APBDEV_PMC_REMOVE_CLAMPING_CMD_0.VE
6. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the VE bit is cleared
7. Clear the following bits to deassert the reset to VI and ISP:
   CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_VI_RST
   CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_ISP_RST
8. If the DIS domain is powered up, deassert the reset to the CSI by clearing the following bit:
   CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_CSI_RST
9. Enable MC clients VI and ISP by clearing the following bits:
   MC_CLIENT_HOTRESET_CTRL_0.VI_FLUSH_ENABLE
   MC_CLIENT_HOTRESET_CTRL_0.ISP_FLUSH_ENABLE

6.8.7.3 Procedures for TD Power Domain

TD Power Gating

1. Flush the NV MC client by setting the following bit:
   MC_CLIENT_HOTRESET_CTRL_0.NV_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:
   NV_HOTRESET_STATUS
3. Set the following bit to assert the reset to 3D:
   CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_3D_RST
4. Clear the following bit to disable clocks to 3D:
   CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_3D
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   PARTID = TD
   START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the TD bit is set

TD Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = TD
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the TD bit is cleared.
3. Set the following bit to enable clocks to 3D:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_3D
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.TD
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the TD bit is cleared.
6. Clear the following bit to deassert the reset to 3D:
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_3D_RST
7. Enable the NV MC client by clearing the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.NV_FLUSH_ENABLE

6.8.7.4 Procedures for VDE Power Domain

VDE Power Gating

1. Flush the VDE MC client by setting the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.VDE_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:
   - VDE_HOTRESET_STATUS
3. Set the following bit to assert the reset to the VDE:
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_VDE_RST
4. Clear the following bit to disable clocks to the VDE:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_VDE
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = VDE
   - START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the VDE bit is set

VDE Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = VDE
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the VDE bit is cleared
3. Set the following bit to enable clocks to the VDE:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_VDE
4. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.VDE
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the VDE bit is cleared
6. Clear the following bit to deassert the reset to the VDE:

[redacted]CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_VDE_RST

7. Enable the VDE MC client by clearing the following bit:

   [redacted]MC_CLIENT_HOTRESET_CTRL_0.VDE_FLUSH_ENABLE

6.8.7.5 Procedures for MPE Power Domain

MPE Power Gating

1. Flush the MSENC MC client by setting the following bit:

   [redacted]MC_CLIENT_HOTRESET_CTRL_0.MSENC_FLUSH_ENABLE

2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit are set:

   [redacted]MSENC_HOTRESET_STATUS

3. Set the following bit to assert the reset to the MSENC:

   [redacted]CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_MSENC_RST

4. Clear the following bit to disable clocks to the MSENC:

   [redacted]CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_MSENC

5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:

   [redacted]PARTID = MPE

   [redacted]START = ENABLE

6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the MPE bit is set

MPE Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:

   [redacted]PARTID = MPE

   [redacted]START = ENABLE

2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the MPE bit is cleared
3. Set the following bits to enable clocks to the MSENC:

   [redacted]CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_MSENC

4. Remove power-gating clamps by writing a 1 to the following bit:

   [redacted]APBDEV_PMC_REMOVE_CLAMPING_CMD_0.MPE

5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the MPE bit is cleared
6. Clear the following bits to deassert the reset to the MSENC:

   [redacted]CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_MSENC_RST

7. Enable the MSENC MC client by clearing the following bit:

   [redacted]MC_CLIENT_HOTRESET_CTRL_0.MSENC_FLUSH_ENABLE

6.8.7.6 Procedures for HEG Power Domain

HEG Power Gating

1. Flush the G2 and EPP MC clients by setting the following bits:

   [redacted]MC_CLIENT_HOTRESET_CTRL_0.G2_FLUSH_ENABLE

   [redacted]MC_CLIENT_HOTRESET_CTRL_0.EPP_FLUSH_ENABLE

2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bits are set:

   [redacted]G2_HOTRESET_STATUS

[redacted]EPP_HOTRESET_STATUS

3. Set the following bits to assert the reset to 2D and EPP:
   [redacted]CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_2D_RST
   [redacted]CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_EPP_RST
4. Clear the following bits to disable clocks to 2D and EPP:
   [redacted]CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_2D
   [redacted]CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_EPP
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   [redacted]PARTID = HEG
   [redacted]START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the HEG bit is set

HEG Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   [redacted]PARTID = HEG
   [redacted]START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the HEG bit is cleared
3. Set the following bits to enable clocks to 2D and EPP:
   [redacted]CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_2D
   [redacted]CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_EPP
4. Remove power-gating clamps by writing a 1 to the following bit:
   [redacted]APBDEV_PMC_REMOVE_CLAMPING_CMD_0.HEG
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the HEG bit is cleared
6. Clear the following bits to deassert the reset to 2D and HEG:
   [redacted]CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_2D_RST
   [redacted]CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_EPP_RST
7. Enable the G2 and EPP MC clients by clearing the following bits:
   [redacted]MC_CLIENT_HOTRESET_CTRL_0.G2_FLUSH_ENABLE
   [redacted]MC_CLIENT_HOTRESET_CTRL_0.EPP_FLUSH_ENABLE

6.8.7.7 Procedures for DIS Power Domain

DIS Power Gating

1. The VE and DISB power domains have to be gated before the DIS can be gated.
2. Flush the DC MC client by setting the following bit:
   [redacted]MC_CLIENT_HOTRESET_CTRL_0.DC_FLUSH_ENABLE
3. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:
   [redacted]DC_HOTRESET_STATUS
4. Set the following bits to assert the reset to DISP1, DSI, DSIB, and MIPI_CAL:
   [redacted]CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_DISP1_RST
   [redacted]CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_DSI_RST
   [redacted]CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_MIPI_CAL_RST

- CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_DSIB_RST

5. Clear the following bits to disable clocks to DISP1, DIS, DISB, and MIPI_CAL:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_DISP1
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_DSI
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_MIPI_CAL
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_DSIB
6. Disable PLLD and PLLD2 by clearing the following bits:
   - CLK_RST_CONTROLLER_PLLD_BASE_0_PLLD_ENABLE
   - CLK_RST_CONTROLLER_PLLD2_BASE_0_PLLD2_ENABLE
7. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = DIS
   - START = ENABLE
8. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the DIS bit is set

DIS Power Ungating

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   - PARTID = DIS
   - START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the DIS bit is cleared
3. If PLLD was enabled before power-gating, enable it by setting the following bit:
   - CLK_RST_CONTROLLER_PLLD_BASE_0_PLLD_ENABLE
4. If PLLD2 was enabled before power-gating, enable it by setting the following bit:
   - CLK_RST_CONTROLLER_PLLD2_BASE_0_PLLD2_ENABLE
5. If either PLLD or PLLD2 was enabled, wait 1 ms for the PLL to lock
6. Set the following bits to enable clocks to DISP1, DSI, DSIB, and MIPI_CAL:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_DISP1
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_DSI
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_MIPI_CAL
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_DSIB
7. If the VE domain is powered up, enable clocks to the CSI by setting the following bit:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_CSI
8. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.DIS
9. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the DIS bit is cleared
10. Write default values to the following registers to clear out any invalid values that might show up after power is ungated:
    - MIPI_CAL_MIPI_CAL_CTRL_0
    - MIPI_CAL_CILA_MIPI_CAL_CONFIG_0
    - MIPI_CAL_CILB_MIPI_CAL_CONFIG_0
    - MIPI_CAL_CILC_MIPI_CAL_CONFIG_0
    - MIPI_CAL_CILD_MIPI_CAL_CONFIG_0
    - MIPI_CAL_CILE_MIPI_CAL_CONFIG_0

- MIPI_CAL_DSIA_MIPI_CAL_CONFIG_0
- MIPI_CAL_DSIB_MIPI_CAL_CONFIG_0
- MIPI_CAL_DSIC_MIPI_CAL_CONFIG_0
- MIPI_CAL_DSID_MIPI_CAL_CONFIG_0

11. Write 1 to the following bits to clear any pending status:
    - MIPI_CAL_CIL_MIPI_CAL_STATUS_0.MIPI_AUTO_CAL_DONE
    - MIPI_CAL_CIL_MIPI_CAL_STATUS_0.MIPI_AUTO_CAL_DONE_CSIA
    - MIPI_CAL_CIL_MIPI_CAL_STATUS_0.MIPI_AUTO_CAL_DONE_CSIB
    - MIPI_CAL_CIL_MIPI_CAL_STATUS_0.MIPI_AUTO_CAL_DONE_CSIC
    - MIPI_CAL_CIL_MIPI_CAL_STATUS_0.MIPI_AUTO_CAL_DONE_CSID
    - MIPI_CAL_CIL_MIPI_CAL_STATUS_0.MIPI_AUTO_CAL_DONE_CSIE
    - MIPI_CAL_CIL_MIPI_CAL_STATUS_0.MIPI_AUTO_CAL_DONE_DSIA
    - MIPI_CAL_CIL_MIPI_CAL_STATUS_0.MIPI_AUTO_CAL_DONE_DSIB
    - MIPI_CAL_CIL_MIPI_CAL_STATUS_0.MIPI_AUTO_CAL_DONE_DSIC
    - MIPI_CAL_CIL_MIPI_CAL_STATUS_0.MIPI_AUTO_CAL_DONE_DSID
12. Read-back MIPI_CAL_CIL_MIPI_CAL_STATUS_0 to push all writes through to the unit.
13. Clear the following bits to deassert the reset to DISP1, DSI, DSIB, and MIPI_CAL:
    - CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_DISP1_RST
    - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_DSI_RST
    - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_MIPI_CAL_RST
    - CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_DSIB_RST
14. If the VE domain is powered up, deassert the reset to the CSI by clearing the following bit:
    - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_CSI_RST
15. Enable the DC MC client by clearing the following bit:
    - MC_CLIENT_HOTRESET_CTRL_0.DC_FLUSH_ENABLE

6.8.7.8 Procedures for DISB Power Domain

DISB Power Gating

1. Flush the DCB MC client by setting the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.DCB_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:
   - DCB_HOTRESET_STATUS
3. Set the following bits to assert the reset to DISP2 and HDMI:
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_DISP2_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_HDMI_RST
4. Clear the following bits to disable clocks to DISP2 and HDMI:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_DISP2
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_HDMI
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:
   - PARTID = DISB
   - START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the DISB bit is set

DISB Power Ungating

1. The DIS power domain has to be ungated before the DISB can be ungated.
2. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:
   - PARTID = DISB
   - START = ENABLE
3. Poll APBDEV_PMC_PWRGATE_STATUS_0 until bit DISB is clear
4. Set the following bits to enable clocks to DISP2 and HDMI:
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0.CLK_ENB_DISP2
   - CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0.CLK_ENB_HDMI
5. Remove power-gating clamps by writing a 1 to the following bit:
   - APBDEV_PMC_REMOVE_CLAMPING_CMD_0.DISB
6. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the DISB bit is clear
7. Clear the following bits to deassert reset to DISP2 and HDMI:
   - CLK_RST_CONTROLLER_RST_DEVICES_L_0.SWR_DISP2_RST
   - CLK_RST_CONTROLLER_RST_DEVICES_H_0.SWR_HDMI_RST
8. Enable the DCB MC client by clearing the following bit:
   - MC_CLIENT_HOTRESET_CTRL_0.DCB_FLUSH_ENABLE

6.8.7.9 Procedures for XUSBA Power Domain

XUSBA Power Gating

1. Set the following bit to assert the reset to XUSB_SS:
   - CLK_RST_CONTROLLER_RST_DEVICES_W_0.SWR_XUSB_SS_RST

2. Clear the following bit to disable clocks to XUSB_SS:

   CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_XUSB_SS
3. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:

   PARTID = XUSBA

   START = ENABLE
4. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the XUSBA bit is set

XUSBA Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:

   PARTID = XUSBA

   START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the XUSBA bit is cleared
3. Set the following bit to enable clocks to XUSB_SS:

   CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0.CLK_ENB_XUSB_SS
4. Remove power-gating clamps by writing a 1 to the following bit:

   APBDEV_PMC_REMOVE_CLAMPING_CMD_0.XUSBA
5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the XUSBA bit is cleared
6. Clear the following bit to deassert the reset to XUSB_SS:

   CLK_RST_CONTROLLER_RST_DEVICES_W_0.SWR_XUSB_SS_RST

6.8.7.10 Procedures for XUSBB Power Domain

XUSBB Power Gating

1. Flush XUSB_DEV MC client by setting the following bit:

   MC_CLIENT_HOTRESET_CTRL_0.XUSB_DEV_FLUSH_ENABLE
2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:

   XUSB_DEV_HOTRESET_STATUS
3. Set the following bits to assert reset to XUSB_DEV:

   CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_XUSB_DEV_RST
4. Clear the following bits to disable clocks to XUSB_DEV:

   CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_XUSB_DEV
5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:

   PARTID = XUSBB

   START = ENABLE
6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the XUSBB bit is set

XUSBB Power Ungating

1. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:

   PARTID = XUSBB

   START = ENABLE
2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the XUSBB bit is clear
3. Set the following bit to enable clocks to XUSB_DEV:

[redacted]CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_XUSB_DEV

4. Remove power-gating clamps by writing a 1 to the following bit:

   [redacted]APBDEV_PMC_REMOVE_CLAMPING_CMD_0.XUSBB

5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the XUSBB bit is cleared
6. Clear the following bit to deassert the reset to XUSB_DEV:

   [redacted]CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_XUSB_DEV_RST

7. Enable the XUSB_DEV MC client by clearing the following bit:

   [redacted]MC_CLIENT_HOTRESET_CTRL_0.XUSB_DEV_FLUSH_ENABLE

6.8.7.11 Procedures for XUSBC Power Domain

XUSBC Power Gating

1. Flush the XUSB_HOST MC client by setting the following bit:

   [redacted]MC_CLIENT_HOTRESET_CTRL_0.XUSB_HOST_FLUSH_ENABLE

2. Poll MC_CLIENT_HOTRESET_STATUS_0 until the following bit is set:

   [redacted]XUSB_HOST_HOTRESET_STATUS

3. Set the following bits to assert the reset to XUSB_HOST:

   [redacted]CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_XUSB_HOST_RST

4. Clear the following bits to disable clocks to XUSB_HOST:

   [redacted]CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_XUSB_HOST

5. Write to these APBDEV_PMC_PWRGATE_TOGGLE_0 fields with the following settings:

   [redacted]PARTID = XUSBC

   [redacted]START = ENABLE

6. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the XUSBC bit is set

XUSBC Power Ungating

1. Write to APBDEV_PMC_PWRGATE_TOGGLE_0 with the following fields:

   [redacted]PARTID = XUSBC

   [redacted]START = ENABLE

2. Poll APBDEV_PMC_PWRGATE_STATUS_0 until the XUSBC bit is cleared
3. Set the following bit to enable clocks to XUSB_HOST:

   [redacted]CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0.CLK_ENB_XUSB_HOST

4. Remove power-gating clamps by writing a 1 to the following bit:

   [redacted]APBDEV_PMC_REMOVE_CLAMPING_CMD_0.XUSBC

5. Poll APBDEV_PMC_REMOVE_CLAMPING_CMD_0 until the XUSBC bit is cleared
6. Clear the following bit to deassert the reset to XUSB_HOST:

   [redacted]CLK_RST_CONTROLLER_RST_DEVICES_U_0.SWR_XUSB_HOST_RST

7. Enable the XUSB_HOST MC client by clearing the following bit:

   [redacted]MC_CLIENT_HOTRESET_CTRL_0.XUSB_HOST_FLUSH_ENABLE

6.9 Clock and Reset Controller Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

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

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

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

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

- ARM7 (COP) and its L1 cache
- All system buses (PPSB, AHB, APB, etc.)
- Timer
- RTC
- NOR Flash controller
- eFUSE
- GPIO
- CoreSight
- Entropy

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

During POR or system reset, the reset controller will deassert reset to the boot blocks first and extend the resets to the MPCore™/ARM7 for another 511 oscillator clock periods. This will prevent either processor from talking to a boot device while it is still in the reset state.

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

6.9.1 Precautions

- Unless noted elsewhere, all modules support changing the clock divider ratio without disabling the clock.
- Unless noted elsewhere, all modules' clock switching is glitch-free except "audio_sync_clk".
- For "audio_sync_clk", the clock source select needs to be set up before changing the device clock source to use that audio clock or to enable the device which use that audio clock.
- Before stopping the clock (via CLK_OUT_ENB_L/H/U/V/W registers) and/or asserting reset (via RST_DEVICES_L/H/U/V/W registers) to a module, first check the module to make sure it is not active. Stopping clock/asserting reset while the module is still busy can cause relatively minor problem such as incorrect data read/written, or catastrophic problem such as system hang. To ensure a module is not active, (a) disable the module by programming its disable bit if not already done so, and (b) wait until the module is not active by checking for its busy bit, done bit, count, or similar mechanism.

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

1. Make sure the device's reset is asserted (via RST_DEVICES_L/H/U/V/W registers).
2. Enable clock to the device (via CLK_OUT_ENB_L/H/U/V/W registers).
3. Change the clock source and/or clock divisor to the device (via CLK_SOURCE_ register).
4. Wait 2 µs to make sure clock source/device logic is stabilized.
5. Deassert the device's reset (via RST_DEVICES_L/H/U/V/W registers).

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

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

1. Make sure the device is disabled (via the device's register).
2. Assert device's reset (via RST_DEVICES_L/H/U/V/W registers).
3. Make sure clock to the device is enabled (via CLK_OUT_ENB_L/H/U/V/W registers).
4. Change the clock source and/or clock divisor to the device (via CLK_SOURCE_<mod> register).
5. Wait 2 µs to make sure clock source/device logic is stabilized.
6. Deassert the device's reset (via RST_DEVICES_L/H/U/V/W registers).

(B) Method 2 -- (not using reset).

1. Make sure the clock to the device is enabled (via CLK_OUT_ENB_L/H/U/V/W registers).
2. Change the clock source and/or the clock divisor to the device (via CLK_SOURCE_<mod> register).
3. Wait 2 µs to make sure the device logic is stabilized.

To reset a device (without need to change clock) after boot-up:

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

Possible Method to wait for 2 µs.

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

Fractional Divide Ratios

Many of the dividers support not only integer divide ratios but also N.5 fractional divide ratios. Special care needs to be exercised when using a fractional divide because the duty cycle of the clock will not be 50/50, which can place a special timing requirement on the logic. From a dynamic voltage scaling (DVS) standpoint, the slightly faster N.0 ratio should be used to determine how low one can lower the voltage.

Invert Duty-Cycle Distortion Control

Some dividers have a feature to help mitigate clock duty-cycle distortion -- the distortion of the positive edge of a clock can be swapped with the distortion of the negative edge. This ability may prove useful to counteract some of the inherent distortion

that occurs in clock generation logic and distribution. For example, a divider with a ratio of 1.0 having a 51%/49% duty cycle could be made to have a 49%/51% duty cycle, if the duty-cycle inversion control is activated.

- **Frequency Change Caution**:

  An INVERT_DCD field can be changed while the clock is running. In circumstances described in item 3 below, this may temporarily have a negative impact on the divided clock frequency, causing the module to fail if the frequency to the module is not temporarily adjusted first.

  1. Asserting INVERT_DCD adds 1/2 clock period (divider input clock) to the low pulse of a single divided output clock period during the change. This causes the divided output clock frequency to be lowered for one clock period.
  2. For ratios other than 1.0, deasserting INVERT_DCD removes 1/2 clock period (divider input clock) from the low pulse of a single divided output clock period during the change. Because this effectively speeds up the divided output clock frequency for one clock period, which can cause a module to fail, the module clock frequency may need to be changed to a lower frequency before deasserting INVERT_DCD.
  3. Deasserting INVERT_DCD with a ratio of 1.0 is unique in that removing 1/2 clock period (divider input clock) from the low pulse effectively removes the low pulse altogether. The resulting combination of the high pulses on either side will effectively look like 1/2 clock period was added to the high pulse instead. Thus there occurs a lower frequency for one clock period rather than an increased frequency like all other divide ratios.

- **Latency**:

  Once the INVERT_DCD signal gets to the divider logic, there will be a maximum delay of 6 divider input clocks + 1 divider output clock before the change will complete.

Main clock sources used by the system:

(A) Primary clocks.

- "osc" or "clk_m" which can be either 12 MHz, 13 MHz, 19.2 MHz 26 MHz, 16.8 MHz, 38.4 MHz, or 48 MHz.
- "clk_s" which is 32 kHz.

(B) PLL clocks.

- "PLLC" is general purpose and its output is called "pllC_out".
- "PLLC2" is general purpose and its output is called "pllC2_out".
- "PLLC3" is general purpose and its output is called "pllC3_out".
- "PLLM" (memory) is primarily used for memory and its output is called "pllM_out".
- "PLLP" (peripheral) has a fixed frequency and its output is called "pllP_out0".
- "PLLA" (audio) is cascaded from PLLP and is used for audio purposes.
- "PLLU" (USB) has fixed outputs at 12 MHz, 48 MHz, 60 MHz, and 480 MHz.
- "PLLD"(DSI) is primarily used for the display and its output (after a /2 fix) is called "pllD_out0".
- "PLLD2"(DSI) is primarily used for the display and its output (after a /2 fix) is called "pllD2_out0".
- "PLLX" has extra high frequency used primarily by the fast CPU and is called "pllX_out0".
- "refPLLE" is only used by PLLE.
- "PLLE" is only used by PCIe, SATA, USB3 (if they are present)

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

- "pllC_out1" is divided-down from "pllC_out0".
- "pllM_out1" is divided-down from "pllM_out0".
- "pllP_out1" is divided-down from "pllP_out0".

- "pllP_out2" is divided-down from "pllP_out0".
- "pllP_out3" is divided-down from "pllP_out0".
- "pllP_out4" is divided-down from "pllP_out0".
- "pllA_out0" is divided-down from the "pllA" output.

Note: With the exception of PLLA and PLLE, the other PLLs all use "osc_div_clk" as the reference clock. This is the same frequency as "osc" with the exception of the following crystals:

38.4 MHz - Needs to be divided by 2 providing 19.2 MHz to the PLLs.

48.0 MHz - Needs to be divided by 4 providing 12.0 MHz to the PLLs

Note: In this document, CPULP, SCPU, and Slow CPU are synonymous. CPUG, FCPU, and Fast CPU are also synonymous.

6.9.2 Multi-Address Tagging

A register or bit marked with a [* Multi-Address] tag indicates multiple addresses can access the same register/bit. The two types of Multi-Address tags are described below.

[CCLK Multi-Address] Tags

These tags apply to legacy, CPUG, and CPULP multiple address registers and bits. For CCLK (CPU_CLK) related registers values:

- Three sets of addresses are provided to support single cluster legacy, CPUG, and CPULP clusters.
- Two sets of hardware flops are used to store programmed values for G and LP CPU clusters. Legacy single cluster registers are aliased to one of these based on the active cluster at the time the register is referenced.
- In some cases, only the CPU bit in an otherwise single access register is multi-address accessible, for example, CLK_ENB_CPU.

[FUSE Multi-Address] Tags

These tags apply to Fuse and CAR multiple address registers:

- A bond_out_* register can be programmed via fuse or CAR CLK_RST_CONTROLLER_BOND_OUT_L_0 register or fuse FUSE_SKU_BOND_OUT_L_0 register.
- Fuse bond_out writes are updated on fuse2all_fuse_outputs_valid rising edge
- CAR bond_out writes are updated on register writes

6.9.3 CLK_RST_CONTROLLER_RST_SOURCE_0

WatchDog (Deadman) Timer

The watchdog timer is used to recover from hang/lockup condition by resetting the system and/or COP (ARM7) and/or CPU (MPCore). Either timer1 or timer2 can be used as watchdog timer. Refer to the Timer section of this document for more information on watchdog timer usage.

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

Bit	R/W	Reset	Description
19	RO	X	WDT_CPU3_RST_STA: CPU3 reset by watchdog timer (RO)

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

6.9.4 CLK_RST_CONTROLLER_RST_DEVICES_L_0

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x7dfffeXX (0b011111x11111111111111111011xxx00x)

Bit	R/W	Reset	Description
31	RW	DISABLE	SWR_CACHE2_RST: Reset COP cache controller. 0 = DISABLE 1 = ENABLE
30	RW	ENABLE	SWR_I2S0_RST: Reset I2S0 Controller 0 = DISABLE 1 = ENABLE
29	RW	ENABLE	SWR_VCP_RST: Reset vector co-processor. 0 = DISABLE 1 = ENABLE
28	RW	ENABLE	SWR_HOST1X_RST: Reset HOST1X. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
27	RW	ENABLE	SWR_DISP1_RST: Reset DISP1 controller. 0 = DISABLE 1 = ENABLE
26	RW	ENABLE	SWR_DISP2_RST: Reset DISP2 controller. 0 = DISABLE 1 = ENABLE
24	RW	ENABLE	SWR_3D_RST: Reset 3D controller. 0 = DISABLE 1 = ENABLE
23	RW	ENABLE	SWR_ISP_RST: Reset ISP controller. 0 = DISABLE 1 = ENABLE
22	RW	ENABLE	SWR_USBD_RST: Reset USB controller. 0 = DISABLE 1 = ENABLE
21	RW	ENABLE	SWR_2D_RST: Reset 2D graphics engine controller. 0 = DISABLE 1 = ENABLE
20	RW	ENABLE	SWR_VI_RST: Reset VI controller. 0 = DISABLE 1 = ENABLE
19	RW	ENABLE	SWR_EPP_RST: Reset EPP controller. 0 = DISABLE 1 = ENABLE
18	RW	ENABLE	SWR_I2S2_RST: Reset I2S 2 Controller 0 = DISABLE 1 = ENABLE
17	RW	ENABLE	SWR_PWM_RST: Reset Pulse Width Modulator 0 = DISABLE 1 = ENABLE
15	RW	ENABLE	SWR_SDMMC4_RST: Reset SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	RW	ENABLE	SWR_SDMMC1_RST: Reset SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	RW	ENABLE	SWR_NDFLASH_RST: Reset NAND flash controller. 0 = DISABLE 1 = ENABLE
12	RW	ENABLE	SWR_I2C1_RST: Reset I2C1 Controller 0 = DISABLE 1 = ENABLE
11	RW	ENABLE	SWR_I2S1_RST: Reset I2S 1 Controller 0 = DISABLE 1 = ENABLE
10	RW	ENABLE	SWR_SPDIF_RST: Reset SPDIF Controller 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
9	RW	ENABLE	SWR_SDMMC2_RST: Reset SDMMC2 Controller 0 = DISABLE 1 = ENABLE
8	RW	DISABLE	SWR_GPIO_RST: Reset GPIO Controller 0 = DISABLE 1 = ENABLE
7	RW	ENABLE	SWR_UARTB_RST: Reset UARTB/VFIR Controller 0 = DISABLE 1 = ENABLE
6	RW	ENABLE	SWR_UARTA_RST: Reset UARTA Controller 0 = DISABLE 1 = ENABLE
5	RO	X	SWR_TMR_RST: Reset Timer Controller 0 = DISABLE 1 = ENABLE
2	RW	DISABLE	SWR_TRIG_SYS_RST: Write 1 to pulse System Reset Signal. Hardware clears this bit. 0 = DISABLE 1 = ENABLE
1	RW	DISABLE	SWR_COP_RST: Write 1 to force COP Reset Signal. Software needs to clear this bit when done. 0 = DISABLE 1 = ENABLE
0	RO	0x1	SWR_CPU_RST: Reserved.

6.9.5 CLK_RST_CONTROLLER_RST_DEVICES_H_0

Offset: 0x8 │ Read/Write: R/W │ Reset: 0xffddf32X (0b1111111111x111x11111x0110x1xx11x)

Bit	R/W	Reset	Description
31	RW	ENABLE	SWR_BSEV_RST: Reset BSEV controller. 0 = DISABLE 1 = ENABLE
30	RW	ENABLE	SWR_BSEA_RST: Reset BSEA controller. 0 = DISABLE 1 = ENABLE
29	RW	ENABLE	SWR_VDE_RST: Reset VDE and BSEV controller. 0 = DISABLE 1 = ENABLE
28	RW	0x1	SWR_MPE_RST: Reserved.
27	RW	ENABLE	SWR_USB3_RST: Reset USB3 controller. 0 = DISABLE 1 = ENABLE
26	RW	ENABLE	SWR_USB2_RST: Reset USB2 controller. 0 = DISABLE 1 = ENABLE
25	RW	ENABLE	SWR_EMC_RST: Reset EMC controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
24	RW	ENABLE	SWR_MIPI_CAL_RST: Reset MIPI CAL Logic. 0 = DISABLE 1 = ENABLE
23	RW	ENABLE	SWR_UARTC_RST: Reset UARTC controller 0 = DISABLE 1 = ENABLE
22	RW	ENABLE	SWR_I2C2_RST: Reset I2C2 controller. 0 = DISABLE 1 = ENABLE
20	RW	ENABLE	SWR_CSI_RST: Reset CSI controller. 0 = DISABLE 1 = ENABLE
19	RW	ENABLE	SWR_HDMI_RST: Reset HDMI 0 = DISABLE 1 = ENABLE
18	RW	ENABLE	SWR_HSI_RST: Reset MIPI base-band HSI controller. 0 = DISABLE 1 = ENABLE
16	RW	ENABLE	SWR_DSI_RST: Reset DSI controller 0 = DISABLE 1 = ENABLE
15	RW	ENABLE	SWR_I2C5_RST: Reset I2C5 controller 0 = DISABLE 1 = ENABLE
14	RW	ENABLE	SWR_SBC3_RST: Reset SBC3 controller. 0 = DISABLE 1 = ENABLE
12	RW	ENABLE	SWR_SBC2_RST: Reset SBC2 controller. 0 = DISABLE 1 = ENABLE
10	RW	DISABLE	SWR_SNOR_RST: Reset NOR Flash controller. 0 = DISABLE 1 = ENABLE
9	RW	ENABLE	SWR_SBC1_RST: Reset SBC1 controller. 0 = DISABLE 1 = ENABLE
8	RW	ENABLE	SWR_KFUSE_RST: Reset KFuse controller. 0 = DISABLE 1 = ENABLE
5	RW	ENABLE	SWR_STAT_MON_RST: Reset statistic monitor. 0 = DISABLE 1 = ENABLE
2	RW	ENABLE	SWR_APBDMA_RST: Reset APB-DMA. 0 = DISABLE 1 = ENABLE
1	RW	ENABLE	SWR_AHBDMA_RST: Reset AHB-DMA. 0 = DISABLE 1 = ENABLE
0	RO	X	SWR_MEM_RST: Reserved.

6.9.6 CLK_RST_CONTROLLER_RST_DEVICES_U_0

Offset: 0xc │ Read/Write: R/W │ Reset: 0x8a8fd5fe (0b1xxx1x1x1xxx11111101010111111111x)

Bit	Reset	Description
31	ENABLE	SWR_XUSB_DEV_RST: Reset XUSB DEV logic. 0 = DISABLE 1 = ENABLE
27	ENABLE	SWR_MSENC_RST: Reset MSENC logic. 0 = DISABLE 1 = ENABLE
25	ENABLE	SWR_XUSB_HOST_RST: Reset XUSB HOST logic. 0 = DISABLE 1 = ENABLE
23	ENABLE	SWR_EMUCIF_RST: Reset EMUCIF logic. 0 = DISABLE 1 = ENABLE
19	ENABLE	SWR_TSEC_RST: Reset TSEC logic. 0 = DISABLE 1 = ENABLE
18	ENABLE	SWR_DSIB_RST: Reset DSIB 0 = DISABLE 1 = ENABLE
17	ENABLE	SWR_I2C_SLOW_RST: Reset I2C_SLOW 0 = DISABLE 1 = ENABLE
16	ENABLE	SWR_NAND_SPEED_RST: Reset NAND_SPEED 0 = DISABLE 1 = ENABLE
15	ENABLE	SWR_DTV_RST: Reset DTV 0 = DISABLE 1 = ENABLE
14	ENABLE	SWR_SOC_THERM_RST: Reset SOC_THERM. This is not implemented in the RTL for security reasons. This field is here to prevent the software common code from faltering. 0 = DISABLE 1 = ENABLE
13	DISABLE	SWR_TRACECLKIN_RST: Reset CoreSight™ traceclkin controller. 0 = DISABLE 1 = ENABLE
11	DISABLE	SWR_AVPUCQ_RST: Reset AVPUCQ logic. 0 = DISABLE 1 = ENABLE
10	ENABLE	SWR_PCIEXCLK_RST: Reset PCIEXCLK logic. 0 = DISABLE 1 = ENABLE
9	DISABLE	SWR_CSITE_RST: Reset CoreSight controller. 0 = DISABLE 1 = ENABLE
8	ENABLE	SWR_AFI_RST: Reset AFI controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	ENABLE	SWR_OWR_RST: Reset OWR controller. 0 = DISABLE 1 = ENABLE
6	ENABLE	SWR_PCIE_RST: Reset PCIe® controller. 0 = DISABLE 1 = ENABLE
5	ENABLE	SWR_SDMMC3_RST: Reset SDMMC3 controller. 0 = DISABLE 1 = ENABLE
4	ENABLE	SWR_SBC4_RST: Reset SBC4 controller. 0 = DISABLE 1 = ENABLE
3	ENABLE	SWR_I2C3_RST: Reset I2C3 controller. 0 = DISABLE 1 = ENABLE
2	0x1	SWR_UARTE_RST: Reserved.
1	ENABLE	SWR_UARTD_RST: Reset UARTD controller. 0 = DISABLE 1 = ENABLE

6.9.7 CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x80000130 (0b100000x00000000000000010011xxx0)

Bit	Reset	Description
31	ENABLE	CLK_ENB_CACHE2: Enable clock to COP cache controller. 0 = DISABLE 1 = ENABLE
30	DISABLE	CLK_ENB_I2S0: Enable clock to I2S0 Controller 0 = DISABLE 1 = ENABLE
29	DISABLE	CLK_ENB_VCP: Enable clock to vector coprocessor. 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_HOST1X: Enable clock to Host1x. 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_DISP1: Enable clock to DISP1 controller. 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_DISP2: Enable clock to DISP2 controller. 0 = DISABLE 1 = ENABLE
24	DISABLE	CLK_ENB_3D: Enable clock to 3D controller. 0 = DISABLE 1 = ENABLE
23	DISABLE	CLK_ENB_ISP: Enable clock to ISP controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	DISABLE	CLK_ENB_USBD: Enable clock to USB controller 0 = DISABLE 1 = ENABLE
21	DISABLE	CLK_ENB_2D: Enable clock to 2D graphics engine. 0 = DISABLE 1 = ENABLE
20	DISABLE	CLK_ENB_VI: Enable clock to VI controller. 0 = DISABLE 1 = ENABLE
19	DISABLE	CLK_ENB_EPP: Enable clock to EPP controller. 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_I2S2: Enable clock to I2S2 controller 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_PWM: Enable clock to PWM (Pulse Width Modulator) 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_SDMMC4: Enable clock to SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_SDMMC1: Enable clock to SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	DISABLE	CLK_ENB_NDFLASH: Enable clock to NAND flash controller. 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_I2C1: Enable clock to I2C1 Controller 0 = DISABLE 1 = ENABLE
11	DISABLE	CLK_ENB_I2S1: Enable clock to I2S1 Controller 0 = DISABLE 1 = ENABLE
10	DISABLE	CLK_ENB_SPDIF: Enable clock to S/PDIF Controller 0 = DISABLE 1 = ENABLE
9	DISABLE	CLK_ENB_SDMMC2: Enable clock to SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	ENABLE	CLK_ENB_GPIO: Enable clock to GPIO Controller 0 = DISABLE 1 = ENABLE
7	DISABLE	CLK_ENB_UARTB: Enable clock to UARTB/VFIR Controller 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_UARTA: Enable clock to UARTA Controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	ENABLE	CLK_ENB_TMR: Enable clock to Timer Controller 0 = DISABLE 1 = ENABLE
4	ENABLE	CLK_ENB_RTC: Enable clock to RTC Controller 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_CPU: Enable clock to CPU. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE

6.9.8 CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000c80 (0b0000000000x000x0000011001000x000)

Bit	Reset	Description
31	DISABLE	CLK_ENB_BSEV: Enable clock to BSEV Controller. 0 = DISABLE 1 = ENABLE
30	DISABLE	CLK_ENB_BSEA: Enable clock to BSEA Controller 0 = DISABLE 1 = ENABLE
29	DISABLE	CLK_ENB_VDE: Enable clock to VDE Controller 0 = DISABLE 1 = ENABLE
28	0x0	CLK_ENB_MPE: Reserved.
27	DISABLE	CLK_ENB_USB3: Enable clock to USB3 controller. 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_USB2: Enable clock to USB2 controller. 0 = DISABLE 1 = ENABLE
25	DISABLE	CLK_ENB_EMC: Enable clock to MC/EMC controller. 0 = DISABLE 1 = ENABLE
24	DISABLE	CLK_ENB_MIPI_CAL: Enable clock to MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	DISABLE	CLK_ENB_UARTC: Enable clock to UARTC controller 0 = DISABLE 1 = ENABLE
22	DISABLE	CLK_ENB_I2C2: Enable clock to I2C2 controller. 0 = DISABLE 1 = ENABLE
20	DISABLE	CLK_ENB_CSI: Enable clock to CSI controller 0 = DISABLE 1 = ENABLE
19	DISABLE	CLK_ENB_HDMI: Enable clock to HDMI 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	DISABLE	CLK_ENB_HSI: Enable clock to MIPI baseband HSI controller. 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_DSI: Enable clock to DSI controller 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_ I2C5: Enable clock to I2C5 controller 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_SBC3: Enable clock to SBC3 controller. 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_SBC2: Enable clock to SBC2 Controller. 0 = DISABLE 1 = ENABLE
11	ENABLE	CLK_ENB_JTAG2TBC: Enable clock to jtag2tbc Interface. 0 = DISABLE 1 = ENABLE
10	ENABLE	CLK_ENB_SNOR: Enable clock to NOR Flash Controller. 0 = DISABLE 1 = ENABLE
9	DISABLE	CLK_ENB_SBC1: Enable clock to SBC1 Controller. 0 = DISABLE 1 = ENABLE
8	DISABLE	CLK_ENB_KFUSE: Enable clock to KFUSE controller. 0 = DISABLE 1 = ENABLE
7	ENABLE	CLK_ENB_FUSE: Enable clock to FUSE controller. 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_PMC: Enable clock to PMC controller. 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_STAT_MON: Enable clock to statistic monitor. 0 = DISABLE 1 = ENABLE
4	DISABLE	CLK_ENB_KBC: Enable clock to keyboard controller. 0 = DISABLE 1 = ENABLE
2	DISABLE	CLK_ENB_APBDMA: Enable clock to APB-DMA. 0 = DISABLE 1 = ENABLE
1	DISABLE	CLK_ENB_AHBDMA: Enable clock to AHB-DMA. 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_MEM: Enable clock to MC. 0 = DISABLE 1 = ENABLE

6.9.9 CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x01f02a00 (0b00000x011111000000101x100000000x)

Bit	Reset	Description
31	DISABLE	CLK_ENB_XUSB_DEV: Enable clock to XUSB DEV. 0 = DISABLE 1 = ENABLE
30	DISABLE	CLK_ENB_DEV1_OUT: Enable clock to DEV1 pad. 0 = DISABLE 1 = ENABLE
29	DISABLE	CLK_ENB_DEV2_OUT: Enable clock to DEV2 pad. 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_SUS_OUT: Enable clock to SUS pad. 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_MSENC: Enable MSENC clk. 0 = DISABLE 1 = ENABLE
25	DISABLE	CLK_ENB_XUSB_HOST: Enable clock to XUSB HOST 0 = DISABLE 1 = ENABLE
24	ENABLE	CLK_ENB_CRAM2: Enable COP cache RAM clock. 0 = DISABLE 1 = ENABLE
23	ENABLE	CLK_ENB_IRAMD: Enable IRAMD clock. 0 = DISABLE 1 = ENABLE
22	ENABLE	CLK_ENB_IRAMC: Enable IRAMC clock. 0 = DISABLE 1 = ENABLE
21	ENABLE	CLK_ENB_IRAMB: Enable IRAMB clock. 0 = DISABLE 1 = ENABLE
20	ENABLE	CLK_ENB_IRAMA: Enable IRAMA clock. 0 = DISABLE 1 = ENABLE
19	DISABLE	CLK_ENB_TSEC: Enable TSEC clock. 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_DSIB: Enable clock to DSIB 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_I2C_SLOW: Enable clock to I2C_SLOW 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_NAND_SPEED: Enable clock to NAND_SPEED 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
15	DISABLE	CLK_ENB_DTV: Enable clock to DTV controller. 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_SOC_THERM: Enable clock to SOC_THERM controller. 0 = DISABLE 1 = ENABLE
13	ENABLE	CLK_ENB_TRACECLKIN: Enable traceclkin to CoreSight. 0 = DISABLE 1 = ENABLE
11	ENABLE	CLK_ENB_AVPUCQ: Enable clock to AVPUCQ. 0 = DISABLE 1 = ENABLE
9	ENABLE	CLK_ENB_CSITE: Enable clock to CoreSight. 0 = DISABLE 1 = ENABLE
8	DISABLE	CLK_ENB_AFI: Enable clock to AFI. 0 = DISABLE 1 = ENABLE
7	DISABLE	CLK_ENB_OWR: Enable clock to OWR. 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_PCIE: Enable clock to PCIe. 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_SDMMC3: Enable clock to SDMMC3. 0 = DISABLE 1 = ENABLE
4	DISABLE	CLK_ENB_SBC4: Enable clock to SBC4. 0 = DISABLE 1 = ENABLE
3	DISABLE	CLK_ENB_I2C3: Enable clock to I2C3. 0 = DISABLE 1 = ENABLE
2	0x0	CLK_ENB_UARTE: Reserved.
1	DISABLE	CLK_ENB_UARTD: Enable clock to UARTD. 0 = DISABLE 1 = ENABLE

6.9.10 CLK_RST_CONTROLLER_CCLK_BURST_POLICY_0

CPU (CCLK) and COP/System (SCLK) Clock Control

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

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

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

- Software can simply keep changing the clock source to one state (i.e., just CWAKEUP_IDLE_SOURCE) without changing the CPU_STATE field.
- Software can have the concept of multiple states by first setting up CWAKEUP__SOURCE and then changing the CPU_STATE field.
- Software can enable hardware auto detect of IRQ/FIQ to jump to the IRQ or FIQ state.

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

CCLK and SCLK Super Clock Divider Control

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

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

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

Note: From a dynamic voltage scaling (DVS) standpoint, the full clock frequency source (not the output frequency of the super clock divider) going into the super clock divider should be used to determine how low one can lower the voltage.

If m > n, the resulting super clock divider output frequency will be the same as the input frequency. In other words, there will be no clock skip or divide down.

Offset: 0x20 │ Read/Write: R/W │ Reset: 0x10X00000 (0b00010000xxxxxxx00000000000000000)

Bit	R/W	Reset	Description
31:28	RW	0x1	CPU_STATE: 0000=32 kHz Clock source; 0001=IDLE Clock Source; 001X=Run clock source; 01XX=IRQ Clock Source; 1XXX=FIQ Clock Source 0 = STDBY 1 = IDLE 2 = RUN 4 = IRQ 8 = FIQ
27	RW	0x0	COP_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1=Burst on COP FIQ
26	RW	0x0	CPU_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1=Burst on CPU FIQ
25	RW	0x0	COP_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1=Burst on COP IRQ
24	RW	0x0	CPU_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1=Burst on CPU IRQ
23	RO	X	TSENSOR_SLOWDOWN: 0 = Normal; 1 = cpug_clk/2 triggered by temperature sensor.
16	RW	0x0	CCLK_RESERVED: Reserved. Only cpulp uses this bit.

Bit	R/W	Reset	Description
15:12	RW	0x0	CWAKEUP_FIQ_SOURCE: 0000 = clk_m, 0001 = pllC_out0, 0010 = clk_s, 0011 = pllM_out0, 0100 = pllP_out0, 0101 = pllP_out4, 0110 = pllC2_out0, 0111 = pllC3_out0, 1000 = PLLX_out0 (low jitter), 1001 = reserved, 1110 = PLLX_out0, (low jitter) 1111 = reserved 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15
11:8	RW	0x0	CWAKEUP_IRQ_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0_LJ 15 = RESERVED15
7:4	RW	0x0	CWAKEUP_RUN_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15
3:0	RW	0x0	CWAKEUP_IDLE_SOURCE: Same definitions as CWAKEUP_FIQ 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15

6.9.11 CLK_RST_CONTROLLER_SUPER_CCLK_DIVIDER_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = Disable divider. 0 = DISABLE 1 = ENABLE
30	DISABLE	SUPER_CDIV_USE_THERM_CONTROLS: 1 = Use therm controls for pulse skipper (cpug only) 0 = DISABLE 1 = ENABLE
28	DISABLE	CCLK_INVERT_DCD: 1 = Enable inversion of DCD (duty-cycle distortion), 0 = Disable inversion of DCD 0 = DISABLE 1 = ENABLE See “Invert Duty-Cycle Distortion Control” in this section for more information.
27	0x0	SUPER_CDIV_DIS_FROM_COP_FIQ: 0 = enable FIQ, disable FIQ.
26	0x0	SUPER_CDIV_DIS_FROM_CPU_FIQ: 0 = enable FIQ, disable FIQ.
25	0x0	SUPER_CDIV_DIS_FROM_COP_IRQ: 0 = enable IRQ, disable IRQ.
24	0x0	SUPER_CDIV_DIS_FROM_CPU_IRQ: 0 = enable IRQ, disable IRQ.
23:16	0x0	CCLK_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_CDIV_DIVISOR: Actual value = n + 1.

6.9.12 CLK_RST_CONTROLLER_SCLK_BURST_POLICY_0

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

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

Bit	Reset	Description
14:12	0x0	SWAKEUP_FIQ_SOURCE: 000 = clk_m, 001 = pllC_out1, 010 = pllP_out4, 011 = pllP_out0, 100 = pllP_out2, 101 = pllC_out, 110 = clk_s, 111 = pllM_out1, 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT0 4 = PLLP_OUT2 5 = CLKD 6 = CLKS 7 = PLLM_OUT1
10:8	0x0	SWAKEUP_IRQ_SOURCE: Same definitions as SWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT0 4 = PLLP_OUT2 5 = CLKD 6 = CLKS 7 = PLLM_OUT1
6:4	0x0	SWAKEUP_RUN_SOURCE: Same definitions as SWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT0 4 = PLLP_OUT2 5 = CLKD 6 = CLKS 7 = PLLM_OUT1
2:0	0x0	SWAKEUP_IDLE_SOURCE: Same definitions as SWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT0 4 = PLLP_OUT2 5 = CLKD 6 = CLKS 7 = PLLM_OUT1

6.9.13 CLK_RST_CONTROLLER_SUPER_SCLK_DIVIDER_0

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

Bit	Reset	Description
31	DISABLE	SUPER_SDIV_ENB: 0 = Disable divider. 0 = DISABLE 1 = ENABLE
27	0x0	SUPER_SDIV_DIS_FROM_COP_FIQ: 0 = enable FIQ, disable FIQ.
26	0x0	SUPER_SDIV_DIS_FROM_CPU_FIQ: 0 = enable FIQ, disable FIQ.
25	0x0	SUPER_SDIV_DIS_FROM_COP_IRQ: 0 = enable IRQ, disable IRQ.
24	0x0	SUPER_SDIV_DIS_FROM_CPU_IRQ: 0 = enable IRQ, disable IRQ.
15:8	0x0	SUPER_SDIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_SDIV_DIVISOR: Actual value = n + 1.

6.9.14 CLK_RST_CONTROLLER_CLK_SYSTEM_RATE_0

Audio Sync Clock

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

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

HCLK/PCLK

SCLK is the main system clock which can run up to 275 MHz.

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

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

Offset: 0x30 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x000x00)

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

6.9.15 CLK_RST_CONTROLLER_COP_CLK_SKIP_POLICY_0

COP Clock Skip

The COP uses the same clock as the system (SCLK). The COP_CLK_SKIP_POLICY register provides a mechanism to slow down the COP without slowing down the system clock. This is done by deferring/delaying the ready signal back to the COP. Again, hardware can auto detect CPU/COP IRQ/FIQ and change the skip rate as programmed by software.

Offset: 0x40 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000xxxxxxxx0000000000000000)

Bit	Reset	Description
31:28	0x0	COP_CLK_SKIP_STATE: 0000=no skip. 0001=skip based on IDLE Clock skip rate; 001X=skip based on Run clock skip rate; 01XX=skip based on IRQ Clock skip rate; 1XXX=skip based on FIQ Clock skip rate
27	0x0	COP_CLK_SKIP_ENB_FROM_COP_FIQ: 0 = NOP ; 1=Burst on COP FIQ
26	0x0	COP_CLK_SKIP_ENB_FROM_CPU_FIQ: 0 = NOP ; 1=Burst on CPU FIQ
25	0x0	COP_CLK_SKIP_ENB_FROM_COP_IRQ: 0 = NOP ; 1=Burst on COP IRQ
24	0x0	COP_CLK_SKIP_ENB_FROM_CPU_IRQ: 0 = NOP ; 1=Burst on CPU IRQ
15:12	0x0	COP_CLK_SKIP_RATE_FIQ: skip n/16 clock.
11:8	0x0	COP_CLK_SKIP_RATE_IRQ: Same definition as COP_CLK_SKIP_RATE_FIQ
7:4	0x0	COP_CLK_SKIP_RATE_RUN: Same definition as COP_CLK_SKIP_RATE_FIQ

Bit	Reset	Description
3:0	0x0	COP_CLK_SKIP_RATE_IDLE: Same definition as COP_CLK_SKIP_RATE_FIQ

6.9.16 CLK_RST_CONTROLLER_CLK_MASK_ARM_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx00xxxxxxxxxxxxxx00)

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

6.9.17 CLK_RST_CONTROLLER_MISC_CLK_ENB_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxxx0000xxxxxxxxxxxxxxxxxxx0)

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

6.9.18 CLK_RST_CONTROLLER_CLK_CPU_CMPLX_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x4c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxx00)

Bit	Reset	Description
11	0x0	CPU3_CLK_STP: 1 = CPU3 clock stop, 0 = CPU3 clock run.
10	0x0	CPU2_CLK_STP: 1 = CPU2 clock stop, 0 = CPU2 clock run.
9	0x0	CPU1_CLK_STP: 1 = CPU1 clock stop, 0 = CPU1 clock run.
8	0x0	CPU0_CLK_STP: 1 = CPU0 clock stop, 0 = CPU0 clock run.
1:0	0x0	CPU_BRIDGE_CLKDIV: Unused but available.

6.9.19 CLK_RST_CONTROLLER_OSC_CTRL_0

Oscillator Control

"osc" can have any of the hardware-supported frequencies (12, 13, 19.2, 26, 16.8, 38.4, 48 MHz) . The OSC_FREQ field provides a way for software to inform hardware what the incoming clock frequency is. This information is used by hardware to auto setup the parameters (DIVN, DIVM, DIVP, CPCON, LFCON, VCOCON, DCCON, OUT1_RATIO, OUT2_RATIO, OUT3_RATIO, and OUT4_RATIO) to PLLP and its dividers. There is a way to override the hardware auto-generated PLLP parameters, if necessary.

Offset: 0x50 │ Read/Write: R/W │ Reset: 0x000003f1 (0b00000000000000x00000xx111111xx01)

Bit	Reset	Description
31:28	0x0	OSC_FREQ: 0000 = 13.0 MHz, 0100 = 19.2 MHz, 1000 = 12.0 MHz, 1100 = 26.00 MHz, 0001 = 16.8 MHz, 0101 = 38.4 MHz*, 1001 = 48.0 MHz*. *Set PLL_REF_DIV to /2 for 38.4 MHz and /4 for 48 MHz. Unused code map to 13 MHz setting in hardware. 0 = OSC13 4 = OSC19P2 8 = OSC12 12 = OSC26 1 = OSC16P8 5 = OSC38P4 9 = OSC48
27:26	0x0	PLL_REF_DIV: PLL reference clock divide. 00 = /1 01 = /2 10 = /4 0 = DIV1 1 = DIV2 2 = DIV4 3 = RESERVED
25:18	0x0	OSCFI_SPARE: Crystal oscillator spare register control.
16:12	0x0	XODS: Crystal oscillator duty cycle control.
9:4	0x3f	XOFS: Crystal oscillator drive strength control.
1	0x0	XOBP: Crystal oscillator bypass enable (1 = enable bypass).
0	0x1	XOE: Crystal oscillator enable (1 = enable).

6.9.20 CLK_RST_CONTROLLER_PLL_LFSR_0

Offset: 0x54 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	RND: Random number generated from PLL linear feedback shift register.

6.9.21 CLK_RST_CONTROLLER_OSC_FREQ_DET_0

Oscillator Frequency Detect

The oscillator can have any one of the seven supported crystal frequencies (12, 13, 19.2, 26, 16.8, 38.4, 48 MHz). Hardware provides the following mechanism to detect the frequency of the oscillator. To determine the oscillator frequency, create an oscillator clock period counting window. The window is defined by programming the number of 32.768 kHz clock periods. Once the counting process is done, the count value can be used to determine the oscillator frequency.

OSC Frequency (MHz)	Approximate OSC_FREQ_DET_CNT (using two 32.768 kHz periods as window)
12.00	732 (0x02DC)
13.00	794 (0x031A)
16.80	1025 (0x0401)
19.20	1172 (0x0494)
26.00	1587 (0x0633)
38.40	2344 (0x0928)
48.00	2930 (0x0B72)

Offset: 0x58 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
31	DISABLE	OSC_FREQ_DET_TRIG: 0 = default, 1 = enable osc frequency detect. 0 = DISABLE 1 = ENABLE
3:0	0x0	REF_CLK_WIN_CFG: Indicates the number of 32.768 kHz clock periods as window in n+1 scheme.

6.9.22 CLK_RST_CONTROLLER_OSC_FREQ_DET_STATUS_0

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

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

6.9.23 CLK_RST_CONTROLLER_PLLC_BASE_0

PLL Configuration

Unless noted otherwise, PLLs are programmed by:

- Setting configuration values.
- Enabling the PLL. It is recommended that the reference clock to the PLL be stable.
- Waiting for the PLL to lock by polling the lock register.

For any PLL specific programming information that may follow, there is an assumption that the default values noted for the register field are already programmed into the registers. Only the changes from these default values are mentioned.

Tegra 4 devices have 12 PLLs controllable by clock control. Consult the PLL specifications for electrical requirements. Below are the PLL cells used for each PLL referenced in this TRM:

- PLLC is of type PLL14G_DYN_ESD.
- PLLC2 and PLLC3 are of type PLL12G_DIG_E1.
- PLLM is of type PLL800_4PH_ESD_C1.

- PLLP and PLLA are of type CLKPLL700_LP_C1.
- PLLU is of type CLKPLL960_USB_B.
- PLLD and PLLD2 are of type CLKPLL10G_MIPI_C.
- PLLX is of type PLL18G_DYN_PRB_ESD.
- refPLLE is of type PLL600_ESD_D1.
- PLLE is of type PLL24G_SSA_33VCML_ESD_C1.

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

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

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

Crystals 38.4 MHz and 48 MHz require pre-divider PLL_REF_DIV settings of 2 and 4, respectively, to meet the input frequency requirements of all PLLs that use "osc_div_clk" as a reference. All other supported crystals are used by the PLLs directly, without any pre-divider.

Unless noted otherwise, writing to the PLL bypass fields can cause clock glitches and a strategy is required to contain these glitches.

Contact your NVIDIA representative for more details on the PLLs.

PLL Post Dividers pll*_out*

These are U7.1 fractional dividers as described earlier. To avoid clock frequencies that the dividers cannot support, they should be placed in reset while the PLL is configured and locking. Only when the PLL clock has locked and is stable should the reset to a divider be deasserted.

The configuration of the divide ratio is glitch-less and can be done at any time. However, if done while the divider in enabled, care should be exercised that driven logic can tolerate both the frequency before the change and the frequency after the change.

PLLP Configuration Information

PLLP is configured as a fixed frequency PLL. The code in the Boot ROM configures PLLP at a fixed 216 MHz. Later on, software (typically the boot loader) may change this to another "fixed" frequency.

For Tegra 4 devices, software changes this frequency from 216 MHz to 408 MHz. Boot ROM configuration values for 216 MHz are listed below.

Table 19: Boot ROM Configuration to 216 MHz

Reference Frequency	13.0 MHz	19.2 MHz	12.0 MHz	26.0 MHz	16.8MHz	38.4MHz	48.0MHz	Resulting Frequency
DIVN	216 (0d8h)	180 (0b4h)	216 (0d8h)	216 (0d8h)	180 (0b4h)	180 (0b4h)	216 (0d8h)	-
DIVM	13 (0dh)	16 (10h)	12 (0ch)	26 (1ah)	14 (0eh)	16 (10h)	12 (0ch)	-
DIVP	1 (0h)	1 (0h)	1 (0h)	1 (0h)	1 (0h)	1 (0h)	1 (0h)	-
CPCON	8 (8h)	4 (4h)	8 (8h)	8 (8h)	4 (4h)	4 (4h)	8 (8h)	-
LFCON	0 (0h)	0 (0h)	0 (0h)	0 (0h)	0 (0h)	0 (0h)	0 (0h)	-
VCOCON	0 (0h)	0 (0h)	0 (0h)	0 (0h)	0 (0h)	0 (0h)	0 (0h)	-
DCCON	0 (0b)	0 (0b)	0 (0b)	0 (0b)	0 (0b)	0 (0b)	0 (0b)	-
PLL_REF_DIV					/1 (0h)	/2 (1h)	/4 (2h)	-

Reference Frequency	13.0 MHz	19.2 MHz	12.0 MHz	26.0 MHz	16.8MHz	38.4MHz	48.0MHz	Resulting Frequency
OUT1 div ratio	7.5 (0dh)	7.5 (0dh)	7.5 (0dh)	7.5 (0dh)	7.5 (0dh)	7.5 (0dh)	7.5 (0dh)	28.8 MHz
OUT2 div ratio	4.5 (07h)	4.5 (07h)	4.5 (07h)	4.5 (07h)	4.5 (07h)	4.5 (07h)	4.5 (07h)	48.0 MHz
OUT3 div ratio	3.0 (04h)	3.0 (04h)	3.0 (04h)	3.0 (04h)	3.0 (04h)	3.0 (04h)	3.0 (04h)	72.0 MHz
OUT4 div ratio	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	2.0 (02h)	108.0 MHz

Table 20: PLLU Configuration Information (Reference Clock osc_div_clk and PLLU-FOs fixed at 12 MHz/48 MHz/60 MHz/480 MHz)

Reference Frequency	13.0 MHz	19.2 MHz	12.0 MHz	26.0 MHz	16.8 MHz	38.4 MHz	48.0 MHz
DIVN	960 (3c0h)	200 (0c8h)	960 (3c0h)	960 (3c0h)	400 (190h)	200 (0c8h)	960 (3c0h)
DIVM	13 (0dh)	4 (04h)	12 (0ch)	26 (1ah)	7 (07h)	4 (04h)	12 (0ch)
CPCON	12 (ch)	3 (3h)	12 (ch)	12 (ch)	5 (5h)	3 (3h)	12 (ch)
LFCON	2 (2h)	2 (2h)	2 (2h)	2 (2h)	2 (2h)	2 (2h)	2 (2h)
PLL_REF_DIV					/1 (0h)	/2 (1h)	/4 (2h)

The settings for PLLP to produce a 408 MHz output are detailed in the next table.

Table 21: PLLP Settings to get 408 MHz Output

Clkin	m	n	p	Update	VCO	Clkout
12	2	68	0	6.00	408	408
12	3	102	0	4.00	408	408
12	4	136	0	3.00	408	408
12	5	170	0	2.40	408	408
12	6	204	0	2.00	408	408
12	7	238	0	1.71	408	408
12	8	272	0	1.50	408	408
12	9	306	0	1.33	408	408
12	10	340	0	1.20	408	408
12	12	408	0	1.00	408	408

Special Consideration when Using PLLX as a Clock Source for the CPU

PLLX is shared between G and LP CPU clusters. To save power, the proper PLLX_FO_LP_DISABLE or PLLX_FO_G_DISABLE bit should only be deasserted when the output is being used.

PLLX source to cpulp (PLLX_OUT0) is DIV2 by default. DIV2 is controlled by PLLX_DIV2_BYPASS_LP bit.

The startup sequence noted in the PLLX datasheet (PLL18G_DYN_PRB_ESD rev 1.6), section 3.4 "PLL Startup sequence" can be simplified to (default register values: PLLX_IDDQ=0, PLLX_ENABLE=0):

1. Program PLL registers (PLLX_ENABLE must be 0 or PLLX_IDDQ must be 1).
2. PLLX_IDDQ 1->0 (if not already 0).
3. Wait 2 µs.
4. PLLX_ENABLE 0->1 (clockin must be running before asserting PLLX_ENABLE). Use PLLX_ENABLE to stop/start the PLL.

Offset: 0x80 | Read/Write: R/W | Reset: 0x0000010c (0bx000xxxx0000xxxx0000000100001100)

Bit	R/W	Reset	Description
30	RW	DISABLE	PLLC_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLC_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE
28	RW	0x0	PLLC_LOCK_OVERRIDE
27	RO	X	PLLC_LOCK: 0 = not lock, 1 = lock.
23:20	RW	0x0	PLLC_DIVP: 0 = post divider (2^n).
15:8	RW	0x1	PLLC_DIVN: PLL feedback divider.
7:0	RW	0xc	PLLC_DIVM: PLL input divider.

6.9.24 CLK_RST_CONTROLLER_PLLC_OUT_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxx00000000xxxxxx11)

Bit	Reset	Description
15:8	0x0	PLLC_OUT1_RATIO: PLLC_OUT1 divider from base PLLC (lsb denoted 0.5x).
1	ENABLE	PLLC_OUT1_CLKEN: PLLC_OUT1 divider clk enable. 0 = DISABLE 1 = ENABLE
0	RESET_DISABLE	PLLC_OUT1_RSTN: PLLC_OUT1 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

6.9.25 CLK_RST_CONTROLLER_PLLC_MISC2_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x0000000X (0bxx00000000000000000000000000000x)

Bit	R/W	Reset	Description
29:28	RW	0x0	PLLC_PTS: PTS field for PLLC. 0 = DISABLE 1 = FO 2 = VCO
27	RW	0x0	PLLC_EN_FSTLCK
26	RW	0x0	PLLC_CLAMP_NDIV
25	RW	0x0	PLLC_EN_DYNRAMP
24:17	RW	0x0	PLLC_DYNRAMP_STEPA
16:9	RW	0x0	PLLC_DYNRAMP_STEPB
8:1	RW	0x0	PLLC_NDIV_NEW
0	RO	X	PLLC_DYNRAMP_DONE

6.9.26 CLK_RST_CONTROLLER_PLLC_MISC_0

Offset: 0x8c │ Read/Write: R/W │ Reset: 0x0X000000 (0b000000x00000000000000000000000)

Bit	R/W	Reset	Description
31	RW	0x0	PLLC_OUT1_INV_CLK: Reserved.
30	RW	0x0	PLLC_OUT1_DIV_BYP: 1 = bypass PLLC_OUT1 divider.
29:28	RW	0x0	PLLC_KCP
27	RW	0x0	PLLC_KVCO
26	RW	0x0	PLLC_IDDQ
25	RO	X	PLLC_FREQ_LOCK
24	RW	0x0	PLLC_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
23:0	RW	0x0	PLLC_SETUP: Base PLLC VCO range setup control.

6.9.27 CLK_RST_CONTROLLER_PLLM_BASE_0

Offset: 0x90 │ Read/Write: R/W │ Reset: 0x00000801 (0b000xxxxxxxxxxx00000100000000001)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLM_BYPASSPLL: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLM_ENABLE: 0 = disable, 1 = enable. (Will invert and connect to IDDQ) 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLM_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLM_LOCK: 0 = not lock, 1 = lock. Phase and frequency
26	RO	X	PLLM_FREQ_LOCK: 0 = not lock, 1 = lock. Frequency lock only
20	RW	0x0	PLLM_DIV2: 1= Frequency of CLKOUT*=VCOCLK/2
19	RW	0x0	PLLM_CLAMP_PX: 1 = clamp CLKOUT_P* / CLKOUT_P*B to 0/1 has priority over IDDQ and ENABLE and PLLAVDD **Note**: PLLM_CLAMP_PX will be cleared before using PLLM. Its purpose is to avoid excessive CDB power consumption while analog PLL power is coming up and the differential output of PLLM is still stabilizing.
15:8	RW	0x8	PLLM_DIVN: PLL feedback divider.
7:0	RW	0x1	PLLM_DIVM: PLL input divider.

6.9.28 CLK_RST_CONTROLLER_PLLM_OUT_0

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

Offset: 0x94 │ Read/Write: R/W │ Reset: 0x00000003 (0bxxxxxxxxxxxxxxx0000000000xxxxxx11)

Bit	Reset	Description
17	0x0	PLLM_OUT1_INV_CLK: Reserved.

Bit	Reset	Description
16	0x0	PLLM_OUT1_DIV_BYP: 1 = bypass PLLM_OUT1 divider
15:8	0x0	PLLM_OUT1_RATIO: PLLM_OUT1 divider from base PLLM (lsb denotes 0.5x).
1	ENABLE	PLLM_OUT1_CLKEN: PLLM_OUT1 divider clk enable. 0 = DISABLE 1 = ENABLE
0	RESET_DISABLE	PLLM_OUT1_RSTN: PLLM_OUT1 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

6.9.29 CLK_RST_CONTROLLER_PLLM_MISC1_0

Offset: 0x98 │ Read/Write: R/W │ Reset: 0x00000000 (0bx00000000000000000000000000000000)

Bit	Reset	Description
30	0x0	PLLM_PD_LSHIFT_PH135
29	0x0	PLLM_PD_LSHIFT_PH90
28	0x0	PLLM_PD_LSHIFT_PH45
27	0x0	PLLM_CLAMP_PH135
26	0x0	PLLM_CLAMP_PH90
25	0x0	PLLM_CLAMP_PH45
24	0x0	PLLM_CLAMP_PH0
23:0	0x0	PLLM_SETUP:SETUP fields

6.9.30 CLK_RST_CONTROLLER_PLLM_MISC2_0

Offset: 0x9c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx000x0000000)

Bit	Reset	Description
10	0x0	PLLM_ENABLE_SW_OVERRIDE: Let software override values on clamp/bypass versus hardware FSM control
9:8	0x0	PLLM_PTS: PTS field for external mux. 0 = DISABLE 1 = FO
6	0x0	PLLM_EN_FSTLCK
5	0x0	PLLM_IDDQ
4	DISABLE	PLLM_PD_LCKDET:1 = Power down lock detect 0 = DISABLE 1 = ENABLE
3	0x0	PLLM_LOCK_OVERRIDE: Forces lock to 1
2:1	0x0	PLLM_KCP: Charge Pump Gain control
0	0x0	PLLM_KVCO:VCO gain

6.9.31 CLK_RST_CONTROLLER_PLLP_BASE_0

Offset: 0xa0 | Read/Write: R/W | Reset: 0x0X00010c (0b0000xxxxx000xx0000000001xxx01100)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLP_BYPASS: 0 = no bypass 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLP_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLP_REF_DIS: 0 = enable reference clock 1 = disable reference clock 0 = REF_ENABLE 1 = REF_DISABLE
28	RW	DISABLE	PLLP_BASE_OVRRIDE: 0 = disallow base override 1 = allow base override 0 = DISABLE 1 = ENABLE
27	RO	X	PLLP_LOCK: 0 = not lock, 1 = lock.
22:20	RW	0x0	PLLP_DIVP: 0 = post divider (2^n).
17:8	RW	0x1	PLLP_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLP_DIVM: PLL input divider.

6.9.32 CLK_RST_CONTROLLER_PLLP_OUTA_0

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00030003 (0b00000000xxxxx01100000000xxxxx011)

Bit	Reset	Description
31:24	0x0	PLLP_OUT2_RATIO: PLLP_OUT2 divider from base PLLP (lsb denotes 0.5x).
18	DISABLE	PLLP_OUT2_OVRRIDE: 0 = disallow PLLP_OUT2 ratio override 1 = enable override 0 = DISABLE 1 = ENABLE
17	ENABLE	PLLP_OUT2_CLKEN: PLLP_OUT2 divider clock enable. 0 = DISABLE 1 = ENABLE
16	RESET_DISABLE	PLLP_OUT2_RSTN: PLLP_OUT2 divider reset. 0 = reset 1 = not reset 0 = RESET_ENABLE 1 = RESET_DISABLE
15:8	0x0	PLLP_OUT1_RATIO: PLLP_OUT1 divider from base PLLP (lsb denotes 0.5x).

Bit	Reset	Description
2	DISABLE	PLLP_OUT1_OVRRIDE: 0 = Disallow PLLP_OUT1 ratio override 1 = enable override 0 = DISABLE 1 = ENABLE
1	ENABLE	PLLP_OUT1_CLKEN: PLLP_OUT1 divider clock enable. 0 = DISABLE 1 = ENABLE
0	RESET_DISABLE	PLLP_OUT1_RSTN: PLLP_OUT1 divider reset. 0 = reset 1 = not reset 0 = RESET_ENABLE 1 = RESET_DISABLE

6.9.33 CLK_RST_CONTROLLER_PLLP_OUTB_0

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

Offset: 0xa8 │ Read/Write: R/W │ Reset: 0x00030003 (0b00000000xxxxx01100000000xxxxx011)

Bit	Reset	Description
31:24	0x0	PLLP_OUT4_RATIO: PLLP_OUT4 divider from base PLLP (lsb denotes 0.5x).
18	DISABLE	PLLP_OUT4_OVRRIDE: 0 = disallow PLLP_OUT4 ratio override 1 = enable override 0 = DISABLE 1 = ENABLE
17	ENABLE	PLLP_OUT4_CLKEN: PLLP_OUT4 divider clock enable. 0 = DISABLE 1 = ENABLE
16	RESET_DISABLE	PLLP_OUT4_RSTN: PLLP_OUT4 divider reset. 0 = reset 1 = not reset 0 = RESET_ENABLE 1 = RESET_DISABLE
15:8	0x0	PLLP_OUT3_RATIO: PLLP_OUT3 divider from base PLLP (lsb denotes 0.5x).
2	DISABLE	PLLP_OUT3_OVRRIDE: 0 = Disallow PLLP_OUT3 ratio override. 0 = DISABLE 1 = ENABLE
1	ENABLE	PLLP_OUT3_CLKEN: PLLP_OUT3 divider clock enable. 0 = DISABLE 1 = ENABLE
0	RESET_DISABLE	PLLP_OUT3_RSTN: PLLP_OUT3 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

6.9.34 CLK_RST_CONTROLLER_PLLP_MISC_0

Offset: 0xac │ Read/Write: R/W │ Reset: 0x00040100 (0b000000000000x1000000000100000000)

Bit	Reset	Description
31	0x0	PLLP_OUT4_INV_CLK: Reserved.
30	0x0	PLLP_OUT3_INV_CLK: Reserved.
29	0x0	PLLP_OUT2_INV_CLK: Reserved.
28	0x0	PLLP_OUT1_INV_CLK: Reserved.
27	0x0	PLLP_OUT4_DIV_BYP: 1 = bypass PLLP_OUT4 divider.
26	0x0	PLLP_OUT3_DIV_BYP: 1 = bypass PLLP_OUT3 divider.
25	0x0	PLLP_OUT2_DIV_BYP: 1 = bypass PLLP_OUT2 divider.
24	0x0	PLLP_OUT1_DIV_BYP: 1 = bypass PLLP_OUT1 divider.
23:22	0x0	PLLP_PTS: Base PLLP test output select. 0 = DISABLE 1 = FO 2 = VCO 3 = NDIV
21	0x0	PLLP_CLKEN_FVCO: FVCO output clock enable
20	0x0	PLLP_DCCON: Base PLLP DCCON control.
18	0x0	PLLP_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
17:12	0x0	PLLP_LOCK_SEL: Lock select.
11:8	0x1	PLLP_CPCON: Base PLLP charge pump setup control.
7:4	0x0	PLLP_LFCON: Base PLLP loop filter setup control.
3:0	0x0	PLLP_VCOCON: Base PLLP VCO range setup control.

6.9.35 CLK_RST_CONTROLLER_PLLA_BASE_0

Offset: 0xb0 │ Read/Write: R/W │ Reset: 0x0X00010c (0b000xxxxxx000xx0000000001xxx01100)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLA_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLA_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLA_REF_DIS: 0 = enable reference clock 1 = disable reference clock 0 = REF_ENABLE 1 = REF_DISABLE

Bit	R/W	Reset	Description
27	RO	X	PLLA_LOCK: 0 = not lock, 1 = lock.
22:20	RW	0x0	PLLA_DIVP: 0 = post divider (2^n).
17:8	RW	0x1	PLLA_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLA_DIVM: PLL input divider.

6.9.36 CLK_RST_CONTROLLER_PLLA_OUT_0

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

Offset: 0xb4 │ Read/Write: R/W │ Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxx00000000xxxxxx11)

Bit	Reset	Description
15:8	0x0	PLLA_OUT0_RATIO: PLLA_OUT0 divider from base PLLA (lsb denotes 0.5x). 0 = DISABLE 1 = ENABLE
1	ENABLE	PLLA_OUT0_CLKEN: PLLA_OUT0 divider clk enable. 0 = DISABLE 1 = ENABLE
0	RESET_DISABLE	PLLA_OUT0_RSTN: PLLA_OUT0 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

6.9.37 CLK_RST_CONTROLLER_PLLA_MISC_0

Offset: 0xbc │ Read/Write: R/W │ Reset: 0x00000100 (0b00xxxxxx0000x00000000000100000000)

Bit	Reset	Description
31	0x0	PLLA_OUT0_INV_CLK: 1 = invert PLLA_OUT0 clock.
30	0x0	PLLA_OUT0_DIV_BYP: 1 = bypass PLLA_OUT0 divider.
23:22	0x0	PLLA_PTS: Base PLLA test output select. 0 = DISABLE 1 = FO 2 = VCO 3 = NDIV
21	0x0	PLLA_CLKEN_FVCO:fvco output clock enable
20	0x0	PLLA_DCCON: Base PLLA DCCON control.
18	0x0	PLLA_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
17:12	0x0	PLLA_LOCK_SEL: Lock select.
11:8	0x1	PLLA_CPCON: Base PLLA charge pump setup control.
7:4	0x0	PLLA_LFCON: Base PLLA loop filter setup control.
3:0	0x0	PLLA_VCOCON: Base PLLA VCO range setup control.

6.9.38 CLK_RST_CONTROLLER_PLLU_BASE_0

Offset: 0xc0 | Read/Write: R/W | Reset: 0x0X00010c (0b000xxx010000xx0000000001xxx01100)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLU_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLU_ENABLE: This bit is used only when the PLLU_OVERRIDE bit is set. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLU_REF_DIS: 0 = enable reference clock 1 = disable reference clock 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLU_LOCK: 0 = not lock, 1 = lock.
25	RW	0x0	PLLU_CLKENABLE_48M: FO_48M output enable. This bit is use only when the PLLU_OVERRIDE bit is set.
23	RW	0x0	PLLU_CLKENABLE_ICUSB: FO_ICUSB output enable. This bit is used only when the PLLU_OVERRIDE bit is set.
22	RW	0x0	PLLU_CLKENABLE_HSIC: FO_HSIC output enable. This bit is used only when the PLLU_OVERRIDE bit is set. Otherwise, USB controllers will control this automatically. 0 = disable 1 = enable
21	RW	0x0	PLLU_CLKENABLE_USB: FO_USB output enable. This bit is used only when the PLLU_OVERRIDE bit is set. Otherwise, USB controllers will control this automatically. 0 = disable 1 = enable
20	RW	0x0	PLLU_VCO_FREQ: 0 = post-divider of 2, 1 = post-divider of 1.
17:8	RW	0x1	PLLU_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLU_DIVM: PLL input divider.

6.9.39 CLK_RST_CONTROLLER_PLLU_MISC_0

Offset: 0xcc | Read/Write: R/W | Reset: 0x00407100 (0bxx000xxxx1xxxx000111000100000000)

Bit	Reset	Description
29:27	0x0	PLLU_PTS: Base PLLU test output select. 0 = DISABLE 1 = VCO 2 = MDIV 3 = NDIV 4 = FO_HSIC 5 = FO_USB 6 = FO_ICUSB 7 = FO_48M

Bit	Reset	Description
22	0x1	PLLU_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
17:12	0x7	PLLU_LOCK_SEL: Lock select.
11:8	0x1	PLLU_CPCON: Base PLLU charge pump setup control.
7:4	0x0	PLLU_LFCON: Base PLLU loop filter setup control.
3:0	0x0	PLLU_VCOCON: Base PLLU VCO range setup control.

6.9.40 CLK_RST_CONTROLLER_PLLD_BASE_0

Offset: 0xd0 │ Read/Write: R/W │ Reset: 0x0X00010c (0b000xx00xx000xx0000000001xxx01100)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLD_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLD_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLD_REF_DIS: 0 = Enable reference clock, 1 = Disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLD_LOCK: 0 = not lock, 1 = lock.
26	RW	0x0	PLLD_CLKENABLE_CSI: 0 = disable, 1 = normal operation. Enable CSI fast and slow P/N clocks to pad macros
25	RW	PLL_D	DSIA_CLK_SRC: 0 = PLL_D, 1 = PLL_D2 0 = PLL_D 1 = PLL_D2
22:20	RW	0x0	PLLD_DIVP: 0 = Post divider (2^n).
17:8	RW	0x1	PLLD_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLD_DIVM: PLL input divider.

6.9.41 CLK_RST_CONTROLLER_PLLD_MISC_0

Offset: 0xdc │ Read/Write: R/W │ Reset: 0x00000100 (0b00000000000000000000000100000000)

Bit	Reset	Description
31	0x0	PLLD_FO_MODE: 1 = 5-125 MHz, 0 = 40-1000 MHz.
30	0x0	PLLD_CLKENABLE: 0 = disable, 1 = normal operation.

Bit	Reset	Description
29:27	0x0	PLLD_PTS: Base PLLD test output select. 0 = DISABLE 1 = DUTYCYCLECTL 2 = VCO 3 = NDIV 4= FO 5 = SLOWCLOCK 6 = FASTCLOCK 7 = LOADPULSE
26:24	0x0	PLLD_LOADADJ: Load pulse position adjust.
23	0x0	PLLD_DIV_RST: 0 = normal operation, 1 = reset.
22	0x0	PLLD_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
21:16	0x0	PLLD_LOCK_SEL: Lock select.
15:12	0x0	PLLD_DCCON: Base PLLD DCCON control.
11:8	0x1	PLLD_CPCON: Base PLLD charge pump setup control.
7:4	0x0	PLLD_LFCON: Base PLLD loop filter setup control.
3:0	0x0	PLLD_VCOCON: Base PLLD VCO range setup control.

6.9.42 CLK_RST_CONTROLLER_PLLX_BASE_0

Offset: 0xe0 │ Read/Write: R/W │ Reset: 0x0X00010c (0b000xxxxx0000xxxx0000000100001100)

Bit	R/W	Reset	Description
31	RW	0x0	PLLX_BYPASS: Reserved.
30	RW	DISABLE	PLLX_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLX_REF_DIS: 0 = Enable reference clock, 1 = Disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLX_LOCK: 0 = Not lock, 1 = Lock.
23:20	RW	0x0	PLLX_DIVP: 0 = Post divider (2^n).
15:8	RW	0x1	PLLX_DIVN: PLL feedback divider.
7:0	RW	0xc	PLLX_DIVM: PLL input divider.

6.9.43 CLK_RST_CONTROLLER_PLLX_MISC_0

Note: PLLX source to cpulp (PLLX_OUT0) is divided by 2 by default. DIV2 is controlled by the PLLX_DIV2_BYPASS_LP bit. See the warning about DIV2 programming in the CLK_RST_CONTROLLER_CCLKLP_BURST_POLICY_0 register.

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0bxx00xxxx00xxx0xxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	0x0	PLLX_FO_LP_DISABLE: PLLX FO_LP output disable. 0=ON, 1=OFF
28	0x0	PLLX_FO_G_DISABLE: PLLX FO_G output disable. 0=ON 1=OFF
23:22	0x0	PLLX_PTS: Base PLLX test output select. 0 = DISABLE 1 = FO 2 = VCO
18	0x0	PLLX_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE

6.9.44 CLK_RST_CONTROLLER_PLLE_BASE_0

Offset: 0xe8 | Read/Write: R/W | Reset: 0x0d00c801 (0bx00011010000000011001000000000001)

Bit	Reset	Description
30	DISABLE	PLLE_ENABLE: PLL enable. 0 = DISABLE 1 = ENABLE
29	0x0	PLLE_LOCK_OVERRIDE: Forces PLL_LOCK and PLL_FREQLOCK to 1.
28	0x0	PLLE_FDIV4B: 0 = vclock/4, 1 = vclock/2 clock to the interpolator logic. Normally set to 0.
27:24	0xd	PLLE_PLDIV_CML: Divider control for CLOCKOUT_CML/CLOCKOUTB_CML.
23:16	0x0	PLLE_EXT_SETUP_23_16: Base PLLE setup [19:16].
15:8	0xc8	PLLE_NDIV: Feedback divider.
7:0	0x1	PLLE_MDIV: Input divider.

6.9.45 CLK_RST_CONTROLLER_PLLE_MISC_0

Offset: 0xec | Read/Write: R/W | Reset: 0x0000XX00 (0b0000000000000000x11xx00000000000)

Bit	R/W	Reset	Description
31:16	RW	0x0	PLLE_SETUP: Base PLLE setup [15:0].
15	RO	X	PLLE_PLL_READY: When read, this bit has PLL_READY status: 1 = PLL finished training, 0 = PLL not finished training.
14	RW	0x1	PLLE_IDDQ_SWCTL: 0 = The PLLE is put in IDDQ mode by hardware (SATA +AFI+CAR) signals.1 = The PLLE is put in IDDQ by software. 0 = OFF 1 = ON(default)
13	RW	0x1	PLLE_IDDQ_OVERRIDE_VALUE: 0 = The PLLE is powered up. 1 = Software can put the PLLE in IDDQ by setting this bit and PLLE_IDDQ_SWCTL 0 = OFF 1 = ON (default)
12	RO	X	PLLE_FREQLOCK: 0 = frequency acquisition not achieved, 1 = frequency acquisition occurred. PLLE_LOCK_ENABLE must be enabled.
11	RO	X	PLLE_LOCK: 0 = not lock (phase+frequency), 1 = lock (phase+frequency). PLLE_LOCK_ENABLE must be enabled.

Bit	R/W	Reset	Description
10	RW	REF_ENABLE	PLLE_REF_DIS: 0 = enable reference clock 1 = disable reference clock 0 = REF_ENABLE 1 = REF_DISABLE
9	RW	0x0	PLLE_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
8	RW	0x0	PLLE_PTS: Bypass PLL (similar to PTO control of other PLLs). 0 = CLOCKOUT*=BYPASSCLK/PLDIV 1 = CLOCKOUT*=VCOCLOCK/PLDIV.
7:6	RW	0x0	PLLE_KCP: Base PLLE charge pump gain control.
5:4	RW	0x0	PLLE_VREG_BG_CTRL: Base PLLE VREG BG control.
3:2	RW	0x0	PLLE_VREG_CTRL: Base PLLE VREG control.
1	RW	DISABLE	PLLE_EN_FSTLCK: 0 = DISABLE 1 = ENABLE
0	RW	0x0	PLLE_KVCO: Base PLLE VCO gain.

6.9.46 CLK_RST_CONTROLLER_PLLS_BASE_0

Offset: 0xf0 │ Read/Write: R/W │ Reset: 0x0X000101 (0b000xxxxxx000xx0000000001xxxx0001)

Bit	R/W	Reset	Description
31	RW	0x0	PLLS_BYPASS: Reserved.
30	RW	0x0	PLLS_ENABLE: Reserved.
29	RW	0x0	PLLS_REF_DIS: Reserved.
27	RO	X	PLLS_LOCK: Reserved.
22:20	RW	0x0	PLLS_DIVP: Reserved.
17:8	RW	0x1	PLLS_DIVN: Reserved.
3:0	RW	0x1	PLLS_DIVM: Reserved.

6.9.47 CLK_RST_CONTROLLER_PLLS_MISC_0

Offset: 0xf4 │ Read/Write: R/W │ Reset: 0x00000100 (0bxxxxxxxxx00xxx0000000000100000000)

Bit	Reset	Description
23:22	0x0	PLLS_PTS: Reserved.
18	0x0	PLLS_LOCK_ENABLE: Reserved.
17:12	0x0	PLLS_LOCK_SEL: Reserved.
11:8	0x1	PLLS_CPCON: Reserved.
7:4	0x0	PLLS_LFCON: Reserved.

Bit	Reset	Description
3:0	0x0	PLLS_VCOCON: Reserved.

6.9.48 CLK_RST_CONTROLLER_CLK_SOURCE_I2S1_0

Clock Configuration for Each Peripheral

In general, each block or module has a dedicated CLK_SOURCE_ register that provides clock source and clock divider control to that device. The clock source select is typically 2 bits long to select one of the four clock sources. The divider is typically 8 bits long, consisting of a 7-bit integer and a 1-bit fraction. Furthermore, depending on the specific module, the CLOCK_SOURCE_ register may contain additional control bits.

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

Special PLL Clock Usage by Certain Peripherals

As mentioned previously, each peripheral has a clock source select in its corresponding CLK_SOURCE_ register. There are a number of modules that take in an additional fixed PLL clock source for portion of their logic.

Listed below are the modules/logic and the fixed PLL clock source they use.

Module/Logic	Fix PLL Clock Source Used
LFSR	pllM_out0
UARTA	pllP_out3
UARTB	pllP_out3
UARTC	pllP_out3
UARTD	pllP_out3
MIPI_CAL	pllP_out3
MIPI_IOBIST	pllP_out3

More information about clock sources and divider widths can be found under “Hardware Features” in this section.

Offset: 0x100 │ Read/Write: R/W │ Reset: 0xd0000000 (0b11x1xxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	I2S1_CLK_SRC: 00 = pllA_out0 01 = audio SYNC_CLK 1x or 2x 10 = pllP_out0 11 = clk_m 0 = PLLA_OUT0 1 = SYNC_CLK_X2 2 = PLLP_OUT0 3 = CLK_M
28	0x1	I2S1_MASTER_CLKEN: Reserved.
7:0	0x0	I2S1_CLK_DIVISOR: N = Divide by (n+1) (lsb denotes 0.5x)

6.9.49 CLK_RST_CONTROLLER_CLK_SOURCE_I2S2_0

Offset: 0x104 | Read/Write: R/W | Reset: 0xd0000000 (0b11x1xxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	I2S2_CLK_SRC: 00 = pllA_out0 01 = audio SYNC_CLK 1x or 2x 10 = pllP_out0 11 = clk_m 0 = PLLA_OUT0 1 = SYNC_CLK_X2 2 = PLLP_OUT0 3 = CLK_M
28	0x1	I2S2_MASTER_CLKEN: Reserved.
7:0	0x0	I2S2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.50 CLK_RST_CONTROLLER_CLK_SOURCE_SPDIF_OUT_0

Offset: 0x108 | Read/Write: R/W | Reset: 0xc0000000 (0b11xxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	SPDIFOUT_CLK_SRC: 00 = pllA_out0 01 = audio SYNC_CLK 1x or 2x 10 = pllP_out0 11 = clk_m 0 = PLLA_OUT0 1 = SYNC_CLK_X2 2 = PLLP_OUT0 3 = CLK_M
7:0	0x0	SPDIF_OUT_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.51 CLK_RST_CONTROLLER_CLK_SOURCE_SPDIF_IN_0

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

Bit	Reset	Description
31:29	PLLP_OUT0	SPDIF_IN_CLK_SRC: 000 = pllP_out0 001 = pllC2_out0 010 = pllC_out0 011 = pllC3_out0 100 = pllM_out0 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0
7:0	0x0	SPDIF_IN_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.52 CLK_RST_CONTROLLER_CLK_SOURCE_PWM_0

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

Bit	Reset	Description
31:29	CLK_M	PWM_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = clk_s, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = CLK_S 6 = CLK_M
7:0	0x0	PWM_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.53 CLK_RST_CONTROLLER_CLK_SOURCE_SBC2_0

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

Bit	Reset	Description
31:29	CLK_M	SBC2_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SBC2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.54 CLK_RST_CONTROLLER_CLK_SOURCE_SBC3_0

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

Bit	Reset	Description
31:29	CLK_M	SBC3_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SBC3_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.55 CLK_RST_CONTROLLER_CLK_SOURCE_I2C1_0

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

Bit	Reset	Description
31:29	CLK_M	I2C1_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
15:0	0x0	I2C1_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 1.0x)

6.9.56 CLK_RST_CONTROLLER_CLK_SOURCE_ I2C5_0

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

Bit	Reset	Description
31:29	CLK_M	I2C5_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
15:0	0x0	I2C5_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 1.0x)

6.9.57 CLK_RST_CONTROLLER_CLK_SOURCE_SBC1_0

Offset: 0x134 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SBC1_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SBC1_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.58 CLK_RST_CONTROLLER_CLK_SOURCE_DISP1_0

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

Bit	Reset	Description
31:29	CLK_M	DISP1_CLK_SRC: 000 = pllP_out0, 010 = pllD_out0, 100 = pllC_out0, 110 = clk_m, 001 = pllM_out0, 011 = pllA_out0, 101 = pllD2_out0, 111 = 1'b0 Non sequential mapping used to preserve 2 MSBs to match legacy source select. 0 = PLLP_OUT0 2 = PLLD_OUT0 4 = PLLC_OUT0 6 = CLK_M 1 = PLLM_OUT0 3 = PLLA_OUT0 5 = PLLD2_OUT0

6.9.59 CLK_RST_CONTROLLER_CLK_SOURCE_DISP2_0

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

Bit	Reset	Description
31:29	CLK_M	DISP2_CLK_SRC: 000 = pllP_out0, 010 = pllD_out0, 100 = pllC_out0, 110 = clk_m, 001 = pllM_out0, 011 = pllA_out0, 101 = pllD2_out0, 111 = 1'b0 Non sequential mapping used to preserve 2 MSBs to match legacy source select. 0 = PLLP_OUT0 2 = PLLD_OUT0 4 = PLLC_OUT0 6 = CLK_M 1 = PLLM_OUT0 3 = PLLA_OUT0 5 = PLLD2_OUT0

6.9.60 CLK_RST_CONTROLLER_CLK_SOURCE_VI_0

Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0b000xxx00xxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLM_OUT0	VI_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 0 = PLLM_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
25	0x0	PD2VI_CLK_SEL: 0 = pd2vi_clk, 1 = vi_sensor_clk.
24	INTERNAL	VI_CLK_SEL: 0 = select internal clock, 1 = select external clock (pd2vi_clk). 0 = INTERNAL 1 = EXTERNAL
7:0	0x0	VI_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.61 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC1_0

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

Bit	Reset	Description
31:29	CLK_M	SDMMC1_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SDMMC1_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.62 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC2_0

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

Bit	Reset	Description
31:29	CLK_M	SDMMC2_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SDMMC2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.63 CLK_RST_CONTROLLER_CLK_SOURCE_G3D_0

Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0b000xx0xxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	PLLM_OUT0	G3D_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 0 = PLLM_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
26	DISABLE	G3D_INVERT_DCD: 1 = Enable inversion of DCD (duty-cycle distortion), 0 = Disable inversion of DCD 0 = DISABLE 1 = ENABLE See “Invert Duty-Cycle Distortion Control” in this section for more information.
15:8	0x0	G3D_IDLE_DIVISOR: N = Divide by (n+1) (lsb denotes 0.5x). If all 0s, this idle divisor field will not be used. For non-zero values, when host1x is idle, this field will be used instead of G3D_CLK_DIVISOR.
7:0	0x0	G3D_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.64 CLK_RST_CONTROLLER_CLK_SOURCE_G2D_0

Offset: 0x15c │ Read/Write: R/W │ Reset: 0x00000000 (0b000xxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	PLLM_OUT0	G2D_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 0 = PLLM_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
15:8	0x0	G2D_IDLE_DIVISOR: N = Divide by (n+1) (lsb denotes 0.5x). If all 0s, this idle divisor field will not be used. For non-zero values, when host1x is idle, this field will be used instead of G2D_CLK_DIVISOR.
7:0	0x0	G2D_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.65 CLK_RST_CONTROLLER_CLK_SOURCE_NDFLASH_0

The Tegra NAND module uses a combination of 3 clocks: nandfst_clk, nand_clk, and nand_speed_clk

- nandfst_clk and nand_clk are controlled by the CLK_SOURCE_NDFLASH register
- nand_speed_clk is controlled by the CLK_SOURCE_NAND_SPEED register

The output of the clock switch is nandfst_clk at full rate.

An additional nand_clk is programmable to full or div2 rate of nandfst_clk.

Offset: 0x160 │ Read/Write: R/W │ Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
31:29	CLK_M	NDFLASH_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
8	0x0	NDFLASH_DIV2_SEL: 0 = ASYNC interface mode. nand_clk = nandfst_clk. 1 = DDR interface mode. nand_clk = nandfst_clk/2
7:0	0x0	NDFLASH_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.66 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC4_0

Offset: 0x164 │ Read/Write: R/W │ Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SDMMC4_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = CLK_M 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M

Bit	Reset	Description
7:0	0x0	SDMMC4_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.67 CLK_RST_CONTROLLER_CLK_SOURCE_VFIR_0

Offset: 0x168 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	VFIR_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	VFIR_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.68 CLK_RST_CONTROLLER_CLK_SOURCE_EPP_0

Offset: 0x16c | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLM_OUT0	EPP_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 0 = PLLM_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
7:0	0x0	EPP_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.69 CLK_RST_CONTROLLER_CLK_SOURCE_MPE_0

Offset: 0x170 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	0x0	MPE_CLK_SRC: Reserved
7:0	0x0	MPE_CLK_DIVISOR: Reserved

6.9.70 CLK_RST_CONTROLLER_CLK_SOURCE_HSI_0

Offset: 0x174 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	HSI_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M

Bit	Reset	Description
7:0	0x0	HSI_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.71 CLK_RST_CONTROLLER_CLK_SOURCE_UARTA_0

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

Bit	Reset	Description
31:29	CLK_M	UARTA_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
24	DISABLE	UARTA_DIV_ENB: UART baud clock divisors by default come directly from the UART DLM/DLL registers in 16.0 format. Enable UART_DIV_ENB to use the 15.1 divisor below. 0=Use UART DLM/DLL, 1=Use divisor below 0 = DISABLE 1 = ENABLE
15:0	0x2	UARTA_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.72 CLK_RST_CONTROLLER_CLK_SOURCE_UARTB_0

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

Bit	Reset	Description
31:29	CLK_M	UARTB_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
24	DISABLE	UARTB_DIV_ENB: UART baud clock divisors by default come directly from UART DLM/DLL registers in 16.0 format. Enable UART_DIV_ENB to use the 15.1 divisor below. 0=Use UART DLM/DLL, 1=Use divisor below 0 = DISABLE 1 = ENABLE
15:0	0x2	UARTB_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.73 CLK_RST_CONTROLLER_CLK_SOURCE_HOST1X_0

Offset: 0x180 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	PLLM_OUT0	HOST1X_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 0 = PLLM_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0

Bit	Reset	Description
15:8	0x0	HOST1X_IDLE_DIVISOR: N = Divide by (n+1) (lsb denotes 0.5x). If all 0s, this idle divisor field will not be used. For non-zero values, when host1x is idle, this field will be used instead of HOST1X_CLK_DIVISOR.
7:0	0x0	HOST1X_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.74 CLK_RST_CONTROLLER_CLK_SOURCE_HDMI_0

Offset: 0x18c │ Read/Write: R/W │ Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

The default for this register is 0xa0000000.

Bit	Reset	Software Default	Description
31:29	CLK_M	PLLD2_OUT0	HDMI_CLK_SRC: 000 = pllP_out0, 010 = pllD_out0, 100 = pllC_out0, 110 = clk_m, 001 = pllM_out0, 011 = pllA_out0, 101 = pllD2_out0, 111 = 1'b0. Non sequential mapping is used to preserve 2 MSBs to match legacy source select. 0 = PLLP_OUT0 2 = PLLD_OUT0 4 = PLLC_OUT0 6 = CLK_M 1 = PLLM_OUT0 3 = PLLA_OUT0 5 = PLLD2_OUT0
7:0	0x0	_NONE_	HDMI_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.75 CLK_RST_CONTROLLER_CLK_SOURCE_I2C2_0

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

Bit	Reset	Description
31:29	CLK_M	I2C2_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
15:0	0x0	I2C2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 1.0x)

6.9.76 CLK_RST_CONTROLLER_CLK_SOURCE_EMC_0

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

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

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

At the end of the sequencing, the DRAMs are restored to being operational. The programming sequence then is:

1. Program the MC/EMC shadow registers corresponding to the new frequency. These registers are "shadowed", as noted in their descriptions, and will not impact the hardware until later during the clock change sequence.
2. Program the EMC clock change FIFO (CCFIFO) with the pre-/post-clk change sequence.

3. Program CAR CLK_SOURCE_EMC register to trigger a clock change. At this point hardware takes over to perform the clock changing sequence.
4. Monitor the clock change complete interrupt status from EMC register to know when clock change is done.

As an alternative to writing one of the sequencing initiating fields, the FORCE_CC_TRIGGER is provided to force the state machine to run. This is not the recommended method as internal restrictions must be met to maintain DRAM contents.

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

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

Offset: 0x19c │ Read/Write: R/W │ Reset: 0x60000000 (0b011x00xxxxxxxxx0xxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	EMC_2X_CLK_SRC: 000 = pllM_out0, 001 = pllC_out0, 010 = pllP_out0, 011 = clk_m, 100 = pllM_out_for_emc undivided PllM_out0, 101 = pllC2_out, 110 = pllC3_out, 111 = empty. 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M 4 = PLLM_UD 5 = PLLC2_OUT0 6 = PLLC3_OUT0
27	0x0	FORCE_CC_TRIGGER: 1 = force a trigger of clock change sequence even if EMC_2X_CLK_SRC/EMC_2X_CLK_DIVISOR fields are unchanged.
26	DISABLE	EMC_INVERT_DCD: 1 = Enable inversion of DCD (duty-cycle distortion), 0 = Disable inversion of DCD. 0 = DISABLE 1 = ENABLE See “Invert Duty-Cycle Distortion Control” in this section for more information.
16	DISABLE	MC_EMC_SAME_FREQ: 0 = MC is 1/2 the frequency of EMC. 1 = MC has the same frequency as the EMC. 0 = DISABLE 1 = ENABLE
7:0	0x0	EMC_2X_CLK_DIVISOR: N = Divide by (n+1) (lsb denotes 0.5x)

6.9.77 CLK_RST_CONTROLLER_CLK_SOURCE_UARTC_0

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

Bit	Reset	Description
31:29	CLK_M	UARTC_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = CLK_M. 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
24	DISABLE	UARTC_DIV_ENB: UART baud clock divisors by default come directly from the UART DLM/DLL registers in 16.0 format. Enable UART_DIV_ENB to use the 15.1 divisor below.0=Use UART DLM/DLL, 1=Use divisor below 0 = DISABLE 1 = ENABLE
15:0	0x2	UARTC_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.78 CLK_RST_CONTROLLER_CLK_SOURCE_VI_SENSOR_0

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLM_OUT0	VI_SENSOR_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 0 = PLLM_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
7:0	0x0	VI_SENSOR_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.79 CLK_RST_CONTROLLER_CLK_SOURCE_SBC4_0

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

Bit	Reset	Description
31:29	CLK_M	SBC4_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SBC4_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.80 CLK_RST_CONTROLLER_CLK_SOURCE_I2C3_0

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

Bit	Reset	Description
31:29	CLK_M	I2C3_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
15:0	0x0	I2C3_CLK_DIVISOR: N = Divide by (n+1) (lsb denotes 1.0x)

6.9.81 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC3_0

Offset: 0x1bc | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SDMMC3_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SDMMC3_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.82 CLK_RST_CONTROLLER_CLK_SOURCE_UARTD_0

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

Bit	Reset	Description
31:29	CLK_M	UARTD_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
24	DISABLE	UARTD_DIV_ENB: UART baud clock divisors by default come directly from the UART DLM/DLL registers in 16.0 format. Enable UART_DIV_ENB to use the 15.1 divisor below. 0=Use UART DLM/DLL, 1=Use divisor below. 0 = DISABLE 1 = ENABLE
15:0	0X2	UARTD_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.83 CLK_RST_CONTROLLER_CLK_SOURCE_UARTE_0

This register is reserved. There is no UART E in the Tegra 4 device.

Offset: 0x1c4 | Read/Write: R/W | Reset: 0xc0000002 (0b110xxxx0xxxxxxxx0000000000000010)

Bit	Reset	Description
31:29	0x6	UARTE_CLK_SRC: Reserved.
24	0x0	UARTE_DIV_ENB: Reserved
15:0	0x2	UARTE_CLK_DIVISOR: Reserved.

6.9.84 CLK_RST_CONTROLLER_CLK_SOURCE_VDE_0

Offset: 0x1c8 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	VDE_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	VDE_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.85 CLK_RST_CONTROLLER_CLK_SOURCE_OWR_0

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

Bit	Reset	Description
31:29	CLK_M	OWR_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	OWR_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.86 CLK_RST_CONTROLLER_CLK_SOURCE_NOR_0

Offset: 0x1d0 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SNOR_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SNOR_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.87 CLK_RST_CONTROLLER_CLK_SOURCE_CSITE_0

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

Bit	Reset	Description
31:29	CLK_M	CSITE_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	CSITE_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.88 CLK_RST_CONTROLLER_CLK_SOURCE_I2S0_0

Offset: 0x1d8 | Read/Write: R/W | Reset: 0xd0000000 (0b11x1xxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	I2S0_CLK_SRC: 00 = pllA_out0, 01 = audio SYNC_CLK 1x or 2x, 10 = pllP_out0, 11 = clk_m 0 = PLLA_OUT0 1 = SYNC_CLK 2 = PLLP_OUT0 3 = CLK_M
28	0x1	I2S0_MASTER_CLKEN: Reserved.
7:0	0x0	I2S0_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.89 CLK_RST_CONTROLLER_CLK_SOURCE_DTV_0

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

Bit	Reset	Description
25	0x0	DTV_INV_CLK: Reserved.

6.9.90 CLK_RST_CONTROLLER_CLK_SOURCE_MSENC_0

Offset: 0x1f0 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLM_OUT0	MSENC_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 0 = PLLM_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
7:0	0x0	MSENC_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.91 CLK_RST_CONTROLLER_CLK_SOURCE_TSEC_0

Offset: 0x1f4 | Read/Write: R/W | Reset: 0x80000000 (0b100xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	PLLM_OUT0	TSEC_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = CLK_M (osc) 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	TSEC_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.92 CLK_RST_CONTROLLER_CLK_SOURCE_OSC_0

Offset: 0x1fc | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28	0x0	OSC_CLK_SRC: Reserved.

6.9.93 CLK_RST_CONTROLLER_CLK_OUT_ENB_X_0

Offset: 0x280 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	CLK_ENB_SPARE: Enable clock - SPARE 0 = DISABLE 1 = ENABLE

6.9.94 CLK_RST_CONTROLLER_CLK_ENB_X_SET_0

Offset: 0x284 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	SET_CLK_ENB_SPARE: Set the enable clock - SPARE 0 = DISABLE 1 = ENABLE

6.9.95 CLK_RST_CONTROLLER_CLK_ENB_X_CLR_0

Offset: 0x288 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CLR_CLK_ENB_SPARE: Clear the enable clock - SPARE 0 = DISABLE 1 = ENABLE

6.9.96 CLK_RST_CONTROLLER_RST_DEVICES_X_0

Offset: 0x28c | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	ENABLE	SWR_SPARE_RST: Reset - SPARE 0 = DISABLE 1 = ENABLE

6.9.97 CLK_RST_CONTROLLER_RST_DEV_X_SET_0

Offset: 0x290 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	SET_SPARE_RST: Set the reset - SPARE 0 = DISABLE 1 = ENABLE

6.9.98 CLK_RST_CONTROLLER_RST_DEV_X_CLR_0

Offset: 0x294 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	CLR_SPARE_RST: Clear the reset - SPARE 0 = DISABLE 1 = ENABLE

6.9.99 CLK_RST_CONTROLLER_DFLL_BASE_0

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

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

6.9.100 CLK_RST_CONTROLLER_RST_DEV_L_SET_0

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

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

Offset: 0x300 | Read/Write: R/W | Reset: 0x7dfffeXX (0b011111x11111111111111111xxx00x)

Bit	R/W	Reset	Description
31	RW	0x0	SET_CACHE2_RST: Set the reset for the COP cache controller. 0 = DISABLE 1 = ENABLE
30	RW	0x1	SET_I2S0_RST: Set the reset for the I2S0 controller. 0 = DISABLE 1 = ENABLE
29	RW	0x1	SET_VCP_RST: Set the reset for the vector coprocessor. 0 = DISABLE 1 = ENABLE
28	RW	0x1	SET_HOST1X_RST: Set the reset for the HOST1X. 0 = DISABLE 1 = ENABLE
27	RW	0x1	SET_DISP1_RST: Set the reset for the DISP1 controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
26	RW	0x1	SET_DISP2_RST: Set the reset for the DISP2 controller. 0 = DISABLE 1 = ENABLE
24	RW	0x1	SET_3D_RST: Set the reset for the 3D controller. 0 = DISABLE 1 = ENABLE
23	RW	0x1	SET_ISP_RST: Set the reset for the ISP controller. 0 = DISABLE 1 = ENABLE
22	RW	0x1	SET_USBD_RST: Set the reset for the USB controller. 0 = DISABLE 1 = ENABLE
21	RW	0x1	SET_2D_RST: Set the reset for the 2D graphics engine controller. 0 = DISABLE 1 = ENABLE
20	RW	0x1	SET_VI_RST: Set the reset for the VI controller. 0 = DISABLE 1 = ENABLE
19	RW	0x1	SET_EPP_RST: Set the reset for the EPP controller. 0 = DISABLE 1 = ENABLE
18	RW	0x1	SET_I2S2_RST: Set the reset for the I2S2 controller. 0 = DISABLE 1 = ENABLE
17	RW	0x1	SET_PWM_RST: Set the reset for the Pulse Width Modulator. 0 = DISABLE 1 = ENABLE
15	RW	0x1	SET_SDMMC4_RST: Set the reset for the SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	RW	0x1	SET_SDMMC1_RST: Set the reset for the SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	RW	0x1	SET_NDFLASH_RST: Set the reset for the NAND Flash controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	SET_I2C1_RST: Set the reset for the I2C1 controller. 0 = DISABLE 1 = ENABLE
11	RW	0x1	SET_I2S1_RST: Set the reset for the I2S1 controller. 0 = DISABLE 1 = ENABLE
10	RW	0x1	SET_SPDIF_RST: Set the reset for the S/PDIF controller. 0 = DISABLE 1 = ENABLE
9	RW	0x1	SET_SDMMC2_RST: Set the reset for the SDMMC2 controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
8	RW	0x0	SET_GPIO_RST: Set the reset for the GPIO controller. 0 = DISABLE 1 = ENABLE
7	RW	0x1	SET_UARTB_RST: Set the reset for the UARTB/VFIR controller. 0 = DISABLE 1 = ENABLE
6	RW	0x1	SET_UARTA_RST: Set the reset for the UARTA controller. 0 = DISABLE 1 = ENABLE
5	RO	x	SET_TMR_RST: Set the reset for the Timer controller. 0 = DISABLE 1 = ENABLE
2	RW	0x0	SET_TRIG_SYS_RST: Write 1 to pulse the System Reset signal. 0 = DISABLE 1 = ENABLE
1	RW	0x0	SET_COP_RST: Set/reset COP. 0 = DISABLE 1 = ENABLE
0	RO	x	SET_CPU_RST: Set the reset for the CPU. [CCLK Multi-Address]. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

6.9.101 CLK_RST_CONTROLLER_RST_DEV_L_CLR_0

Offset: 0x304 | Read/Write: R/W | Reset: 0x7dfffeXX (0b011111x11111111111111111011xxxx0x)

Bit	R/W	Reset	Description
31	RW	0x0	CLR_CACHE2_RST: Clear the reset for the COP cache controller. 0 = DISABLE 1 = ENABLE
30	RW	0x1	CLR_I2S0_RST: Clear the reset for the I2S0 controller. 0 = DISABLE 1 = ENABLE
29	RW	0x1	CLR_VCP_RST: Clear the reset for the vector coprocessor. 0 = DISABLE 1 = ENABLE
28	RW	0x1	CLR_HOST1X_RST: Clear the reset for the HOST1X. 0 = DISABLE 1 = ENABLE
27	RW	0x1	CLR_DISP1_RST: Clear the reset for the DISP1 controller. 0 = DISABLE 1 = ENABLE
26	RW	0x1	CLR_DISP2_RST: Clear the reset for the DISP2 controller. 0 = DISABLE 1 = ENABLE.
24	RW	0x1	CLR_3D_RST: Clear the reset for the 3D controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
23	RW	0x1	CLR_ISP_RST: Clear the reset for the ISP controller. 0 = DISABLE 1 = ENABLE
22	RW	0x1	CLR_USBD_RST: Clear the reset for the USB controller. 0 = DISABLE 1 = ENABLE
21	RW	0x1	CLR_2D_RST: Clear the reset for the 2D graphics engine controller. 0 = DISABLE 1 = ENABLE
20	RW	0x1	CLR_VI_RST: Clear the reset for the VI controller. 0 = DISABLE 1 = ENABLE
19	RW	0x1	CLR_EPP_RST: Clear the reset for the EPP controller. 0 = DISABLE 1 = ENABLE
18	RW	0x1	CLR_I2S2_RST: Clear the reset for the I2S 2 controller. 0 = DISABLE 1 = ENABLE
17	RW	0x1	CLR_PWM_RST: Clear the reset for the Pulse Width Modulator. 0 = DISABLE 1 = ENABLE
15	RW	0x1	CLR_SDMMC4_RST: Clear the reset for the SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	RW	0x1	CLR_SDMMC1_RST: Clear the reset for the SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	RW	0x1	CLR_NDFLASH_RST: Clear the reset for the NAND Flash controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	CLR_I2C1_RST: Clear the reset for the I2C1 controller. 0 = DISABLE 1 = ENABLE
11	RW	0x1	CLR_I2S1_RST: Clear the reset for the I2S1 controller. 0 = DISABLE 1 = ENABLE
10	RW	0x1	CLR_SPDIF_RST: Clear the reset for the S/PDIF controller. 0 = DISABLE 1 = ENABLE
9	RW	0x1	CLR_SDMMC2_RST: Clear the reset for the SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	RW	0x0	CLR_GPIO_RST: Clear the reset for the GPIO controller. 0 = DISABLE 1 = ENABLE
7	RW	0x1	CLR_UARTB_RST: Clear the reset for the UARTB/VFIR controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
6	RW	0x1	CLR_UARTA_RST: Clear the reset for the UARTA controller. 0 = DISABLE 1 = ENABLE
5	RO	X	CLR_TMR_RST: Clear the reset for the Timer controller. 0 = DISABLE 1 = ENABLE
1	RW	0x0	CLR_COP_RST: Clear the reset for the COP. 0 = DISABLE 1 = ENABLE
0	RW	X	CLR_CPU_RST: Clear the reset for the CPU. [CCLK Multi-Address] This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

6.9.102 CLK_RST_CONTROLLER_RST_DEV_H_SET_0

Offset: 0x308 │ Read/Write: R/W │ Reset: 0xffddf32X (0b1111111111x111x11111x0110x1xx11x)

Bit	R/W	Reset	Description
31	RW	0x1	SET_BSEV_RST: Set the reset for the BSEV controller. 0 = DISABLE 1 = ENABLE
30	RW	0x1	SET_BSEA_RST: Set the reset for the BSEA controller. 0 = DISABLE 1 = ENABLE
29	RW	0x1	SET_VDE_RST: Set the reset for the VDE controller. 0 = DISABLE 1 = ENABLE
28	RW	0x1	SET_MPE_RST: Reserved.
27	RW	0x1	SET_USB3_RST: Set the reset for the USB3 controller. 0 = DISABLE 1 = ENABLE
26	RW	0x1	SET_USB2_RST: Set the reset for the USB2 controller. 0 = DISABLE 1 = ENABLE
25	RW	0x1	SET_EMC_RST: Set the reset for the EMC controller. 0 = DISABLE 1 = ENABLE
24	RW	0x1	SET_MIPI_CAL_RST: Set the reset for the MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	RW	0x1	SET_UARTC_RST: Set the reset for the UARTC controller. 0 = DISABLE 1 = ENABLE
22	RW	0x1	SET_I2C2_RST: Set the reset for the I2C2 controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
20	RW	0x1	SET_CSI_RST: Set the reset for the CSI controller. 0 = DISABLE 1 = ENABLE
19	RW	0x1	SET_HDMI_RST: Set the reset for the HDMI 0 = DISABLE 1 = ENABLE
18	RW	0x1	SET_HSI_RST: Set the reset for the MIPI baseband HSI controller. 0 = DISABLE 1 = ENABLE
16	RW	0x1	SET_DSI_RST: Set the reset for the DSI controller. 0 = DISABLE 1 = ENABLE
15	RW	0x1	SET_ I2C5_RST: Set the reset for the I2C5 controller. 0 = DISABLE 1 = ENABLE
14	RW	0x1	SET_SBC3_RST: Set the reset for the SBC3 controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	SET_SBC2_RST: Set the reset for the SBC2 controller. 0 = DISABLE 1 = ENABLE
10	RW	0x0	SET_SNOR_RST: Set the reset for the NOR Flash controller. 0 = DISABLE 1 = ENABLE
9	RW	0x1	SET_SBC1_RST: Set the reset for the SBC1 Controller. 0 = DISABLE 1 = ENABLE
8	RW	0x1	SET_KFUSE_RST: Set the reset for the KFuse controller. 0 = DISABLE 1 = ENABLE
5	RW	0x1	SET_STAT_MON_RST: Set the reset for the statistic monitor 0 = DISABLE 1 = ENABLE
2	RW	0x1	SET_APBDMA_RST: Set the reset for the APB-DMA. 0 = DISABLE 1 = ENABLE
1	RW	0x1	SET_AHBDMA_RST: Set the reset for the AHB-DMA. 0 = DISABLE 1 = ENABLE
0	RO	X	SET_MEM_RST: Set/reset MC. This bit is disabled. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

6.9.103 CLK_RST_CONTROLLER_RST_DEV_H_CLR_0

Offset: 0x30c | Read/Write: R/W | Reset: 0xffddf32X (0b1111111111x111x11111x0110x1xx11x)

Bit	R/W	Reset	Description
31	RW	0x1	CLR_BSEV_RST: Clear the reset for the BSEV controller. 0 = DISABLE 1 = ENABLE
30	RW	0x1	CLR_BSEA_RST: Clear the reset for the BSEA controller. 0 = DISABLE 1 = ENABLE
29	RW	0x1	CLR_VDE_RST: Clear the reset for the VDE controller. 0 = DISABLE 1 = ENABLE
28	RW	0x1	CLR_MPE_RST: Reserved.
27	RW	0x1	CLR_USB3_RST: Clear the reset for the USB3 controller. 0 = DISABLE 1 = ENABLE
26	RW	0x1	CLR_USB2_RST: Clear the reset for the USB2 controller. 0 = DISABLE 1 = ENABLE
25	RW	0x1	CLR_EMC_RST: Clear the reset for the EMC controller. 0 = DISABLE 1 = ENABLE
24	RW	0X1	CLR_MIPI_CAL_RST: Clear the reset for the MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	RW	0x1	CLR_UARTC_RST: Clear the reset for the UARTC controller. 0 = DISABLE 1 = ENABLE
22	RW	0x1	CLR_I2C2_RST: Clear the reset for the I2C2 controller. 0 = DISABLE 1 = ENABLE
20	RW	0x1	CLR_CSI_RST: Clear the reset for the CSI controller. 0 = DISABLE 1 = ENABLE
19	RW	0x1	CLR_HDMI_RST: Clear the reset for the HDMI. 0 = DISABLE 1 = ENABLE
18	RW	0x1	CLR_MIPI_RST: Clear the reset for the MIPI baseband HSI controller. 0 = DISABLE 1 = ENABLE
16	RW	0x1	CLR_DSI_RST: Clear the reset for the DSI controller. 0 = DISABLE 1 = ENABLE
15	RW	0x1	CLR_ I2C5_RST: Clear the reset for the I2C5 controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
14	RW	0x1	CLR_SBC3_RST: Clear the reset for the SBC3 controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	CLR_SBC2_RST: Clear the reset for the SBC2 controller. 0 = DISABLE 1 = ENABLE
10	RW	0x0	CLR_SNOR_RST: Clear the reset for the NOR Flash controller. 0 = DISABLE 1 = ENABLE
9	RW	0x1	CLR_SBC1_RST: Clear the reset for the SBC1 controller. 0 = DISABLE 1 = ENABLE
8	RW	0x1	CLR_KFUSE_RST: Clear the reset for the KFuse controller. 0 = DISABLE 1 = ENABLE
5	RW	0x1	CLR_STAT_MON_RST: Clear the reset for the statistic monitor. 0 = DISABLE 1 = ENABLE
2	RW	0x1	CLR_APBDMA_RST: Clear the reset for the APB-DMA. 0 = DISABLE 1 = ENABLE
1	RW	0x1	CLR_AHBDMA_RST: Clear the reset for the AHB-DMA. 0 = DISABLE 1 = ENABLE
0	RO	X	CLR_MEM_RST: Clear the reset for the MC. This bit is disabled. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

6.9.104 CLK_RST_CONTROLLER_RST_DEV_U_SET_0

Offset: 0x310 │ Read/Write: R/W │ Reset: 0x8a8fd5fe (0b1xxx1x1x1xxx11111101010111111111x)

Bit	Reset	Description
31	0x1	SET_XUSB_DEV_RST: Set the reset for the XUSB DEV logic. 0 = DISABLE 1 = ENABLE
27	0x1	SET_MSENC_RST: Set the reset for the MSENC logic. 0 = DISABLE 1 = ENABLE
25	0x1	SET_XUSB_HOST_RST: Set the reset for the XUSB HOST logic. 0 = DISABLE 1 = ENABLE
23	0x1	SET_EMUCIF_RST: Set the reset for the EMUCIF logic 0 = DISABLE 1 = ENABLE
19	0x1	SET_TSEC_RST: Set the reset for the TSEC logic 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	0x1	SET_DSIB_RST: Set the reset for the DSIB controller 0 = DISABLE 1 = ENABLE
17	0x1	SET_I2C_SLOW_RST: Set the reset for the I2C_SLOW 0 = DISABLE 1 = ENABLE
16	0x1	SET_NAND_SPEED_RST: Set the reset for the NAND_SPEED. 0 = DISABLE 1 = ENABLE
15	0x1	SET_DTV_RST: Set the reset for the DTV controller. 0 = DISABLE 1 = ENABLE
14	0x1	SET_SOC_THERM_RST: Set the reset for the SOC_THERM controller. 0 = DISABLE 1 = ENABLE
13	0x0	SET_TRACECLKIN_RST: Set the reset for the TRACECLKIN controller. 0 = DISABLE 1 = ENABLE
11	0x0	SET_AVPUCQ_RST: Set the reset for the AVPUCQ logic. 0 = DISABLE 1 = ENABLE
10	0x1	SET_PCIEXCLK_RST: Set the reset for the PCIEXCLK logic. 0 = DISABLE 1 = ENABLE
9	0x0	SET_CSITE_RST: Set the reset for the CSITE controller. 0 = DISABLE 1 = ENABLE
8	0x1	SET_AFI_RST: Set the reset for the AFI controller. 0 = DISABLE 1 = ENABLE
7	0x1	SET_OWR_RST: Set the reset for the OWR controller. 0 = DISABLE 1 = ENABLE
6	0x1	SET_PCIE_RST: Set the reset for the PCIe controller. 0 = DISABLE 1 = ENABLE
5	0x1	SET_SDMMC3_RST: Set the reset for the SDMMC3 controller. 0 = DISABLE 1 = ENABLE
4	0x1	SET_SBC4_RST: Set the reset for the SBC4 controller. 0 = DISABLE 1 = ENABLE
3	0x1	SET_I2C3_RST: Set the reset for the I2C3 controller. 0 = DISABLE 1 = ENABLE
2	0x1	SET_UARTE_RST: Reserved.

Bit	Reset	Description
1	0x1	SET_UARTD_RST: Set the reset for the UARTD controller. 0 = DISABLE 1 = ENABLE

6.9.105 CLK_RST_CONTROLLER_RST_DEV_U_CLR_0

Offset: 0x314 │ Read/Write: R/W │ Reset: 0x8a8fd5fe (0b1xxx1x1x1xxx11111101010111111111x)

Bit	Reset	Description
31	0x1	CLR_XUSB_DEV_RST: Clear the reset for the XUSB DEV logic. 0 = DISABLE 1 = ENABLE
27	0x1	CLR_MSENC_RST: Clear the reset for the MSENC logic 0 = DISABLE 1 = ENABLE
25	0x1	CLR_XUSB_HOST_RST: Clear the reset for the XUSB HOST logic. 0 = DISABLE 1 = ENABLE
23	0x1	CLR_EMUCIF_RST: Clear the reset for the EMUCIF logic 0 = DISABLE 1 = ENABLE
19	0x1	CLR_TSEC_RST: Clear the reset for the TSEC logic 0 = DISABLE 1 = ENABLE
18	0x1	CLR_DSIB_RST: Clear the reset for the DSIB Controller 0 = DISABLE 1 = ENABLE
17	0x1	CLR_I2C_SLOW_RST: Clear the reset for the I2C_SLOW 0 = DISABLE 1 = ENABLE
16	0x1	CLR_NAND_SPEED_RST: Clear the reset for the NAND_SPEED 0 = DISABLE 1 = ENABLE
15	0x1	CLR_DTV_RST: Clear the reset for the DTV controller 0 = DISABLE 1 = ENABLE
14	0x1	CLR_SOC_THERM_RST: Clear the reset for the SOC_THERM controller. 0 = DISABLE 1 = ENABLE
13	0x0	CLR_TRACECLKIN_RST: Clear the reset for the TRACECLKIN controller. 0 = DISABLE 1 = ENABLE
11	0x0	CLR_AVPUCQ_RST: Clear the reset for the AVPUCQ logic. 0 = DISABLE 1 = ENABLE
10	0x1	CLR_PCIEXCLK_RST: Clear the reset for the PCIEXCLK logic. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	0x0	CLR_CSITE_RST: Clear the reset for the CSITE controller. 0 = DISABLE 1 = ENABLE
8	0x1	CLR_AFI_RST: Clear the reset for the AFI controller. 0 = DISABLE 1 = ENABLE
7	0x1	CLR_OWR_RST: Clear the reset for the OWR controller. 0 = DISABLE 1 = ENABLE
6	0x1	CLR_PCIE_RST: Clear the reset for the PCIe controller. 0 = DISABLE 1 = ENABLE
5	0x1	CLR_SDMMC3_RST: Clear the reset for the SDMMC3 controller. 0 = DISABLE 1 = ENABLE
4	0x1	CLR_SBC4_RST: Clear the reset for the SBC4 controller. 0 = DISABLE 1 = ENABLE
3	0x1	CLR_I2C3_RST: Clear the reset for the I2C3 controller. 0 = DISABLE 1 = ENABLE
2	0x1	CLR_UARTE_RST: Reserved.
1	0x1	CLR_UARTD_RST: Clear the reset for the UARTD controller. 0 = DISABLE 1 = ENABLE

6.9.106 CLK_RST_CONTROLLER_CLK_ENB_L_SET_0

Offset: 0x320 │ Read/Write: R/W │ Reset: 0x80000130 (0b100000x00000000000000010011xxx0)

Bit	Reset	Description
31	0x1	SET_CLK_ENB_CACHE2: Set the enable clock to COP cache controller. 0 = DISABLE 1 = ENABLE
30	0x0	SET_CLK_ENB_I2S0: Set the enable clock to I2S0 controller 0 = DISABLE 1 = ENABLE
29	0x0	SET_CLK_ENB_VCP: Set the enable clock to vector coprocessor. 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_HOST1X: Set the enable clock to HOST1X. 0 = DISABLE 1 = ENABLE
27	0x0	SET_CLK_ENB_DISP1: Set the enable clock to DISP1 controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26	0x0	SET_CLK_ENB_DISP2: Set the enable clock to DISP2 controller. 0 = DISABLE 1 = ENABLE
24	0x0	SET_CLK_ENB_3D: Set the enable clock to 3D controller. 0 = DISABLE 1 = ENABLE
23	0x0	SET_CLK_ENB_ISP: Set the enable clock to ISP controller. 0 = DISABLE 1 = ENABLE
22	0x0	SET_CLK_ENB_USBD: Set the enable clock to USB controller 0 = DISABLE 1 = ENABLE
21	0x0	SET_CLK_ENB_2D: Set the enable clock to 2D graphics engine. 0 = DISABLE 1 = ENABLE
20	0x0	SET_CLK_ENB_VI: Set the enable clock to VI controller. 0 = DISABLE 1 = ENABLE
19	0x0	SET_CLK_ENB_EPP: Set the enable clock to EPP controller. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CLK_ENB_I2S2: Set the enable clock to I2S2 controller 0 = DISABLE 1 = ENABLE
17	0x0	SET_CLK_ENB_PWM: Set the enable clock to PWM (Pulse Width Modulator) 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_SDMMC4: Set the enable clock to SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	0x0	SET_CLK_ENB_SDMMC1: Set the enable clock to SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	0x0	SET_CLK_ENB_NDFLASH: Set the enable clock to NAND Flash controller. 0 = DISABLE 1 = ENABLE
12	0x0	SET_CLK_ENB_I2C1: Set the enable clock to I2C1 controller 0 = DISABLE 1 = ENABLE
11	0x0	SET_CLK_ENB_I2S1: Set the enable clock to I2S1 controller 0 = DISABLE 1 = ENABLE
10	0x0	SET_CLK_ENB_SPDIF: Set the enable clock to S/PDIF controller 0 = DISABLE 1 = ENABLE
9	0x0	SET_CLK_ENB_SDMMC2: Set the enable clock to SDMMC2 controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x1	SET_CLK_ENB_GPIO: Set the enable clock to GPIO Controller 0 = DISABLE 1 = ENABLE
7	0x0	SET_CLK_ENB_UARTB: Set the enable clock to UARTB/VFIR controller 0 = DISABLE 1 = ENABLE
6	0x0	SET_CLK_ENB_UARTA: Set the enable clock to UARTA controller 0 = DISABLE 1 = ENABLE
5	0x1	SET_CLK_ENB_TMR: Set the enable clock to Timer controller 0 = DISABLE 1 = ENABLE
4	0x1	SET_CLK_ENB_RTC: Set the enable clock to RTC controller 0 = DISABLE 1 = ENABLE
0	0x0	SET_CLK_ENB_CPU: Set the enable clock to CPU. [CCLK Multi-Address 0 = DISABLE 1 = ENABLE

6.9.107 CLK_RST_CONTROLLER_CLK_ENB_L_CLR_0

Offset: 0x324 │ Read/Write: R/W │ Reset: 0x80000130 (0b100000x00000000000000010011xxx0)

Bit	Reset	Description
31	0x1	CLR_CLK_ENB_CACHE2: Clear the enable clock to COP cache controller. 0 = DISABLE 1 = ENABLE
30	0x0	CLR_CLK_ENB_I2S0: Clear the enable clock to I2S0 controller 0 = DISABLE 1 = ENABLE
29	0x0	CLR_CLK_ENB_VCP: Clear the enable clock to vector coprocessor. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_HOST1X: Clear the enable clock to HOST1X. 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_DISP1: Clear the enable clock to DISP1 controller. 0 = DISABLE 1 = ENABLE
26	0x0	CLR_CLK_ENB_DISP2: Clear the enable clock to DISP2 controller. 0 = DISABLE 1 = ENABLE
24	0x0	CLR_CLK_ENB_3D: Clear the enable clock to 3D controller. 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CLK_ENB_ISP: Clear the enable clock to ISP controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	0x0	CLR_CLK_ENB_USBD: Clear the enable clock to USB controller 0 = DISABLE 1 = ENABLE
21	0x0	CLR_CLK_ENB_2D: Clear the enable clock to 2D graphics engine. 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CLK_ENB_VI: Clear the enable clock to VI controller. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CLK_ENB_EPP: Clear the enable clock to EPP controller. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_I2S2: Clear the enable clock to I2S2 controller 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CLK_ENB_PWM: Clear the enable clock to PWM (Pulse Width Modulator) 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_SDMMC4: Clear the enable clock to SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	0x0	CLR_CLK_ENB_SDMMC1: Clear the enable clock to SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	0x0	CLR_CLK_ENB_NDFLASH: Clear the enable clock to NAND Flash controller. 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_I2C1: Clear the enable clock to I2C1 controller 0 = DISABLE 1 = ENABLE
11	0x0	CLR_CLK_ENB_I2S1: Clear the enable clock to I2S1 controller 0 = DISABLE 1 = ENABLE
10	0x0	CLR_CLK_ENB_SPDIF: Clear the enable clock to S/PDIF controller 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CLK_ENB_SDMMC2: Clear the enable clock to SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	0x1	CLR_CLK_ENB_GPIO: Clear the enable clock to GPIO controller 0 = DISABLE 1 = ENABLE
7	0x0	CLR_CLK_ENB_UARTB: Clear the enable clock to UARTB/VFIR controller 0 = DISABLE 1 = ENABLE
6	0x0	CLR_CLK_ENB_UARTA: Clear the enable clock to UARTA controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x1	CLR_CLK_ENB_TMR: Clear the enable clock to Timer controller 0 = DISABLE 1 = ENABLE
4	0x1	CLR_CLK_ENB_RTC: Clear the enable clock to RTC controller 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CLK_ENB_CPU: Clear the enable clock to CPU. 0 = DISABLE 1 = ENABLE

6.9.108 CLK_RST_CONTROLLER_CLK_ENB_H_SET_0

Offset: 0x328 │ Read/Write: R/W │ Reset: 0x00000480 (0b0000000000x000x00000x1001000x000)

Bit	Reset	Description
31	0x0	SET_CLK_ENB_BSEV: Set the enable clock to BSEV controller. 0 = DISABLE 1 = ENABLE
30	0x0	SET_CLK_ENB_BSEA: Set the enable clock to BSEA controller 0 = DISABLE 1 = ENABLE
29	0x0	SET_CLK_ENB_VDE: Set the enable clock to VDE controller 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_MPE: Reserved.
27	0x0	SET_CLK_ENB_USB3: Set the enable clock to USB3 controller. 0 = DISABLE 1 = ENABLE
26	0x0	SET_CLK_ENB_USB2: Set the enable clock to USB2 controller. 0 = DISABLE 1 = ENABLE
25	0x0	SET_CLK_ENB_EMC: Set the enable clock to MC/EMC controller. 0 = DISABLE 1 = ENABLE
24	0x0	SET_CLK_ENB_MIPI_CAL: Set the enable clock to MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	0x0	SET_CLK_ENB_UARTC: Set the enable clock to UARTC controller 0 = DISABLE 1 = ENABLE
22	0x0	SET_CLK_ENB_I2C2: Set the enable clock to I2C2 controller. 0 = DISABLE 1 = ENABLE
20	0x0	SET_CLK_ENB_CSI: Set the enable clock to CSI controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19	0x0	SET_CLK_ENB_HDMI: Set the enable clock to HDMI. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CLK_ENB_HSI: Set the enable clock to MIPI baseband HSI controller. 0 = DISABLE 1 = ENABLE
16	0x0	SET_CLK_ENB_DSI: Set the enable clock to DSI controller. 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_ I2C5: Set the enable clock to I2C5 controller. 0 = DISABLE 1 = ENABLE
14	0x0	SET_CLK_ENB_SBC3: Set the enable clock to SBC3 (SPI 3) controller. 0 = DISABLE 1 = ENABLE
12	0x0	SET_CLK_ENB_SBC2: Set the enable clock to SBC2 (SPI 2) controller. 0 = DISABLE 1 = ENABLE
10	0x1	SET_CLK_ENB_SNOR: Set the enable clock to NOR Flash controller. 0 = DISABLE 1 = ENABLE
9	0x0	SET_CLK_ENB_SBC1: Set the enable clock to SBC1 controller. 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_KFUSE: Set the enable clock to KFUSE controller. 0 = DISABLE 1 = ENABLE
7	0x1	SET_CLK_ENB_FUSE: Set the enable clock to FUSE controller. 0 = DISABLE 1 = ENABLE
6	0x0	SET_CLK_ENB_PMC: Set the enable clock to PMC controller. 0 = DISABLE 1 = ENABLE
5	0x0	SET_CLK_ENB_STAT_MON: Set the enable clock to statistic monitor. 0 = DISABLE 1 = ENABLE
4	0x0	SET_CLK_ENB_KBC: Set the enable clock to keyboard controller. 0 = DISABLE 1 = ENABLE
2	0x0	SET_CLK_ENB_APBDMA: Set the enable clock to APB-DMA. 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_AHBDMA: Set the enable clock to AHB-DMA. 0 = DISABLE 1 = ENABLE
0	0x0	SET_CLK_ENB_MEM: Set the enable clock to MC. 0 = DISABLE 1 = ENABLE

6.9.109 CLK_RST_CONTROLLER_CLK_ENB_H_CLR_0

Offset: 0x32c │ Read/Write: R/W │ Reset: 0x00000480 (0b0000000000x000x00000x1001000x000)

Bit	Reset	Description
31	0x0	CLR_CLK_ENB_BSEV: Clear the enable clock to BSEV controller. 0 = DISABLE 1 = ENABLE
30	0x0	CLR_CLK_ENB_BSEA: Clear the enable clock to BSEA controller 0 = DISABLE 1 = ENABLE
29	0x0	CLR_CLK_ENB_VDE: Clear the enable clock to VDE controller 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_MPE: Reserved.
27	0x0	CLR_CLK_ENB_USB3: Clear the enable clock to USB3 controller. 0 = DISABLE 1 = ENABLE
26	0x0	CLR_CLK_ENB_USB2: Clear the enable clock to USB2 controller. 0 = DISABLE 1 = ENABLE
25	0x0	CLR_CLK_ENB_EMC: Clear the enable clock to MC/EMC controller. 0 = DISABLE 1 = ENABLE
24	0x0	CLR_CLK_ENB_MIPI_CAL: Clear the enable clock to MIPI CAL logic. 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CLK_ENB_UARTC: Clear the enable clock to UARTC controller. 0 = DISABLE 1 = ENABLE
22	0x0	CLR_CLK_ENB_I2C2: Clear the enable clock to I2C2 controller. 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CLK_ENB_CSI: Clear the enable clock to CSI controller. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CLK_ENB_HDMI: Clear the enable clock to HDMI. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_HSI: Clear the enable clock to MIPI base-band HSI controller. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CLK_ENB_DSI: Clear the enable clock to DSI controller. 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_ I2C5: Clear the enable clock to -I2C5 controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
14	0x0	CLR_CLK_ENB_SBC3: Clear the enable clock to SBC3 controller. 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_SBC2: Clear the enable clock to SBC 2 Controller. 0 = DISABLE 1 = ENABLE
10	0x1	CLR_CLK_ENB_SNOR: Clear the enable clock to NOR Flash Controller. 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CLK_ENB_SBC1: Clear the enable clock to SBC 1 Controller. 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CLK_ENB_KFUSE: Clear the enable clock to KFUSE controller. 0 = DISABLE 1 = ENABLE
7	0x1	CLR_CLK_ENB_FUSE: Clear the enable clock to FUSE controller. 0 = DISABLE 1 = ENABLE
6	0x0	CLR_CLK_ENB_PMC: Clear the enable clock to PMC controller. 0 = DISABLE 1 = ENABLE
5	0x0	CLR_CLK_ENB_STAT_MON: Clear the enable clock to statistic monitor. 0 = DISABLE 1 = ENABLE
4	0x0	CLR_CLK_ENB_KBC: Clear the enable clock to keyboard controller. 0 = DISABLE 1 = ENABLE
2	0x0	CLR_CLK_ENB_APBDMA: Clear the enable clock to APB-DMA. 0 = DISABLE 1 = ENABLE
1	0x0	CLR_CLK_ENB_AHBDMA: Clear the enable clock to AHB-DMA. 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CLK_ENB_MEM: Clear the enable clock to MC 0 = DISABLE 1 = ENABLE

6.9.110 CLK_RST_CONTROLLER_CLK_ENB_U_SET_0

Offset: 0x330 │ Read/Write: R/W │ Reset: 0x09f82a00 (0b00001x011111100000101x100000000x)

Bit	Reset	Description
31	0x0	SET_CLK_ENB_XUSB_DEV: Set the enable clock to XUSB DEV. 0 = DISABLE 1 = ENABLE
30	0x0	SET_CLK_ENB_DEV1_OUT: Set the enable clock to DEV1 pad. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
29	0x0	SET_CLK_ENB_DEV2_OUT: Set the enable clock to DEV2 pad. 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_SUS_OUT: Set the enable clock to SUS pad. 0 = DISABLE 1 = ENABLE
27	0x1	SET_CLK_ENB_MSENC: Set the enable clock to the MSENC clock. 0 = DISABLE 1 = ENABLE
25	0x0	SET_CLK_ENB_XUSB_HOST: Set the enable clock to XUSB HOST. 0 = DISABLE 1 = ENABLE
24	0x1	SET_CLK_ENB_CRAM2: Set the enable clock for the COP cache RAM clock. 0 = DISABLE 1 = ENABLE
23	0x1	SET_CLK_ENB_IRAMD: Set the enable clock for the IRAMD clock. 0 = DISABLE 1 = ENABLE
22	0x1	SET_CLK_ENB_IRAMC: Set the enable clock for the IRAMC clock. 0 = DISABLE 1 = ENABLE
21	0x1	SET_CLK_ENB_IRAMB: Set the enable clock for the IRAMB clock. 0 = DISABLE 1 = ENABLE
20	0x1	SET_CLK_ENB_IRAMA: Set the enable clock for the IRAMB clock. 0 = DISABLE 1 = ENABLE
19	0x1	SET_CLK_ENB_TSEC: Set the enable clock for the TSEC clock. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CLK_ENB_DSIB: Set the enable clock to DSIB controller 0 = DISABLE 1 = ENABLE
17	0x0	SET_CLK_ENB_I2C_SLOW: Set the enable clock to I2C_SLOW 0 = DISABLE 1 = ENABLE
16	0x0	SET_CLK_ENB_NAND_SPEED: Set the enable clock to NAND_SPEED 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_DTV: Set the enable clock to DTV Controller. Moved to U:15 0 = DISABLE 1 = ENABLE
14	0x0	SET_CLK_ENB_SOC_THERM: Set the enable clock to SOC_THERM controller. 0 = DISABLE 1 = ENABLE
13	0x1	SET_CLK_ENB_TRACECLKIN: Set the enable clock to TRACECLKIN. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x1	SET_CLK_ENB_AVPUCQ: Set the enable clock to AVPUCQ. 0 = DISABLE 1 = ENABLE
9	0x1	SET_CLK_ENB_CSITE: Set the enable clock to CSITE. 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_AFI: Set the enable clock to AFI. 0 = DISABLE 1 = ENABLE
7	0x0	SET_CLK_ENB_OWR: Set the enable clock to OWR. 0 = DISABLE 1 = ENABLE
6	0x0	SET_CLK_ENB_PCIE: Set the enable clock to PCIe. 0 = DISABLE 1 = ENABLE
5	0x0	SET_CLK_ENB_SDMMC3: Set the enable clock to SDMMC3. 0 = DISABLE 1 = ENABLE
4	0x0	SET_CLK_ENB_SBC4: Set the enable clock to SBC4. 0 = DISABLE 1 = ENABLE
3	0x0	SET_CLK_ENB_I2C3: Set the enable clock to I2C3. 0 = DISABLE 1 = ENABLE
2	0x0	SET_CLK_ENB_UARTE: Reserved.
1	0x0	SET_CLK_ENB_UARTD: Set the enable clock to UARTD. 0 = DISABLE 1 = ENABLE

6.9.111 CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0

Offset: 0x334 │ Read/Write: R/W │ Reset: 0x01f02a00 (0b00000x011111000000101x100000000x)

Bit	Reset	Description
31	0x0	CLR_CLK_ENB_XUSB_DEV: Clear the enable clock to XUSB DEV. 0 = DISABLE 1 = ENABLE
30	0x0	CLR_CLK_ENB_DEV1_OUT: Clear the enable clock to DEV1 pad. 0 = DISABLE 1 = ENABLE
29	0x0	CLR_CLK_ENB_DEV2_OUT: Clear the enable clock to DEV2 pad. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_SUS_OUT: Clear the enable clock to SUS pad. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	0x0	CLR_CLK_ENB_MSENC: Clear the enable clock to the MSENC clock. 0 = DISABLE 1 = ENABLE
25	0x0	CLR_CLK_ENB_XUSB_HOST: Clear the enable clock to XUSB HOST. 0 = DISABLE 1 = ENABLE
24	0x1	CLR_CLK_ENB_CRAM2: Clear the enable clock to the COP cache RAM clock. 0 = DISABLE 1 = ENABLE
23	0x1	CLR_CLK_ENB_IRAMD: Clear the enable clock to the IRAMD clock. 0 = DISABLE 1 = ENABLE
22	0x1	CLR_CLK_ENB_IRAMC: Clear the enable clock to the IRAMC clock. 0 = DISABLE 1 = ENABLE
21	0x1	CLR_CLK_ENB_IRAMB: Clear the enable clock to the IRAMB clock. 0 = DISABLE 1 = ENABLE
20	0x1	CLR_CLK_ENB_IRAMA: Clear the enable clock to the IRAMB clock. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CLK_ENB_TSEC: Clear the enable clock to the TSEC clock. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_DSIB: Clear the enable clock to DSIB controller. 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CLK_ENB_I2C_SLOW: Clear the enable clock to I2C_SLOW 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CLK_ENB_NAND_SPEED: Clear the enable clock to NAND_SPEED 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_DTV: Clear the enable clock to DTV controller. 0 = DISABLE 1 = ENABLE
14	0x0	CLR_CLK_ENB_SOC_THERM: Clear the enable clock to SOC_THERM controller. 0 = DISABLE 1 = ENABLE
13	0x1	CLR_CLK_ENB_TRACECLKIN: Clear the enable clock to TRACECLKIN. 0 = DISABLE 1 = ENABLE
11	0x1	CLR_CLK_ENB_AVPUCQ: Clear the enable clock to AVPUCQ. 0 = DISABLE 1 = ENABLE
9	0x1	CLR_CLK_ENB_CSITE: Clear the enable clock to CSITE. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	CLR_CLK_ENB_AFI: Clear the enable clock to AFI. 0 = DISABLE 1 = ENABLE
7	0x0	CLR_CLK_ENB_OWR: Clear the enable clock to OWR. 0 = DISABLE 1 = ENABLE
6	0x0	CLR_CLK_ENB_PCIE: Clear the enable clock to PCIe. 0 = DISABLE 1 = ENABLE
5	0x0	CLR_CLK_ENB_SDMMC3: Clear the enable clock to SDMMC3. 0 = DISABLE 1 = ENABLE
4	0x0	CLR_CLK_ENB_SBC4: Clear the enable clock to SBC4. 0 = DISABLE 1 = ENABLE
3	0x0	CLR_CLK_ENB_I2C3: Clear the enable clock to I2C3. 0 = DISABLE 1 = ENABLE
2	0x0	CLR_CLK_ENB_UARTE: Reserved.
1	0x0	CLR_CLK_ENB_UARTD: Clear the enable clock to UARTD. 0 = DISABLE 1 = ENABLE

6.9.112 CLK_RST_CONTROLLER_CCPLEX_PG_SM_OVRD_0

CCPLEX Power Gate State-Machine Overrides

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

Offset: 0x33c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23	0x0	CLKSTOP_OVRD_SCPU_NC: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
22	0x0	CLKSTOP_OVRD_SCPU_0: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
21	0x0	CLKSTOP_OVRD_FCPU_RAIL: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
20	0x0	CLKSTOP_OVRD_FCPU_NC: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
19	0x0	CLKSTOP_OVRD_FCPU_3: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
18	0x0	CLKSTOP_OVRD_FCPU_2: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
17	0x0	CLKSTOP_OVRD_FCPU_1: 1 = Clamp the corresponding clocks

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
16	0x0	CLKSTOP_OVRD_FCPU_0: 1 = Clamp the corresponding clocks 0 = DISABLE 1 = ENABLE
15	0x0	RST_OVRD_SCPU_NC: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
14	0x0	RST_OVRD_SCPU_0: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
13	0x0	RST_OVRD_FCPU_RAIL: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
12	0x0	RST_OVRD_FCPU_NC: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
11	0x0	RST_OVRD_FCPU_3: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
10	0x0	RST_OVRD_FCPU_2: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
9	0x0	RST_OVRD_FCPU_1: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
8	0x0	RST_OVRD_FCPU_0: 1 = Assert the corresponding reset 0 = DISABLE 1 = ENABLE
7	0x0	EN_RST_CG_OVRD_SCPU_NC: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
6	0x0	EN_RST_CG_OVRD_SCPU_0: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
5	0x0	EN_RST_CG_OVRD_FCPU_RAIL: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
4	0x0	EN_RST_CG_OVRD_FCPU_NC: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
3	0x0	EN_RST_CG_OVRD_FCPU_3: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
2	0x0	EN_RST_CG_OVRD_FCPU_2: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
1	0x0	EN_RST_CG_OVRD_FCPU_1: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE
0	0x0	EN_RST_CG_OVRD_FCPU_0: 1 = Select the OVRD controls in this register 0 = DISABLE 1 = ENABLE

6.9.113 CLK_RST_CONTROLLER_RST_CPU_CMPLX_SET_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

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

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

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

6.9.114 CLK_RST_CONTROLLER_RST_CPU_CMPLX_CLR_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Offset: 0x344 │ Read/Write: R/W │ Reset: 0x2000feef (0bx010xxx00000000011111110111011111)

Bit	Reset	Description
30	0x0	CLR_PRESETDBG: 1 = Deassert nPRESETDBG to the CoreSight. 0 = DISABLE 1 = ENABLE
29	0x1	CLR_NONCPURESET: 1 = Deassert reset to the whole nonCPU region of the CPU. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_PERIPHRESET: Reserved.

Bit	Reset	Description
24	0x0	CLR_L2RESET: 1 = Deassert nL2RESET to CPU. 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CXRESET3: 1 = Deassert nCXRESET to CPU3. N/A in CPULP. 0 = DISABLE 1 = ENABLE
22	0x0	CLR_CXRESET2: 1 = Deassert nCXRESET to CPU2. N/A in CPULP. 0 = DISABLE 1 = ENABLE
21	0x0	CLR_CXRESET1: 1 = Deassert nCXRESET to CPU1. N/A in CPULP. 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CXRESET0: 1 = Deassert nCXRESET to CPU0. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CORERESET3: 1 = Deassert nCORERESET to CPU3. N/A in CPULP. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CORERESET2: 1 = Deassert nCORERESET to CPU2. N/A in CPULP. 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CORERESET1: 1 = Deassert nCORERESET to CPU1. N/A in CPULP. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CORERESET0: 1 = Deassert nCORERESET to CPU0. 0 = DISABLE 1 = ENABLE
15	0x1	CLR_DBGRESET3: 1 = Deassert nDBGRESET to CPU3. N/A in CPULP. 0 = DISABLE 1 = ENABLE
14	0x1	CLR_DBGRESET2: 1 = Deassert nDBGRESET to CPU2. N/A in CPULP. 0 = DISABLE 1 = ENABLE
13	0x1	CLR_DBGRESET1: 1 = Deassert nDBGRESET to CPU1. N/A in CPULP. 0 = DISABLE 1 = ENABLE
12	0x1	CLR_DBGRESET0: 1 = Deassert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
11	0x1	CLR_WDRESET3: Reserved.
10	0x1	CLR_WDRESET2: Reserved.
9	0x1	CLR_WDRESET1: Reserved.
8	0x0	CLR_WDRESET0: Reserved.
7	0x1	CLR_DERESET3: Reserved.
6	0x1	CLR_DERESET2: Reserved.
5	0x1	CLR_DERESET1: Reserved.

Bit	Reset	Description
4	0x0	CLR_DERESET0: Reserved.
3	0x1	CLR_CPURESET3: 1 = Deassert nCPUPORRESET to CPU3. N/A in CPULP. 0 = DISABLE 1 = ENABLE
2	0x1	CLR_CPURESET2: 1 = Deassert nCPUPORRESET to CPU2. N/A in CPULP. 0 = DISABLE 1 = ENABLE
1	0x1	CLR_CPURESET1: 1 = Deassert nCPUPORRESET to CPU1. N/A in CPULP. 0 = DISABLE 1 = ENABLE
0	0x1	CLR_CPURESET0: 1 = Deassert nCPUPORRESET to CPU0. 0 = DISABLE 1 = ENABLE

6.9.115 CLK_RST_CONTROLLER_CLK_CPU_CMPLX_SET_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x348 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxxx)

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

6.9.116 CLK_RST_CONTROLLER_CLK_CPU_CMPLX_CLR_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x34c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxxxx)

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

6.9.117 CLK_RST_CONTROLLER_RST_DEVICES_V_0

Expanded Register Set

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

Offset: 0x358 │ Read/Write: R/W │ Reset: 0xff81ffeX (0b111111111xxxxxx111111111111x11xx)

Bit	R/W	Reset	Description
29	RW	ENABLE	SWR_HDA_RST: Reset High Definition Audio 0 = DISABLE 1 = ENABLE
28	RW	ENABLE	SWR_SATA_RST: Reset SATA 0 = DISABLE 1 = ENABLE
23	RW	ENABLE	SWR_ACTMON_RST: Reset ACTMON 0 = DISABLE 1 = ENABLE
16	RW	ENABLE	SWR_ATOMICS_RST: Reset ATOMICS 0 = DISABLE 1 = ENABLE
15	RW	ENABLE	SWR_HDA2CODEC_2X_RST: Reset HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	RW	ENABLE	SWR_DAM2_RST: Reset DAM2 0 = DISABLE 1 = ENABLE
13	RW	ENABLE	SWR_DAM1_RST: Reset DAM1 0 = DISABLE 1 = ENABLE
12	RW	ENABLE	SWR_DAM0_RST: Reset DAM0 0 = DISABLE 1 = ENABLE
11	RW	ENABLE	SWR_APBIF_RST: Reset APBIF 0 = DISABLE 1 = ENABLE
10	RW	ENABLE	SWR_AUDIO_RST: Reset AUDIO 0 = DISABLE 1 = ENABLE
9	RW	ENABLE	SWR_SBC6_RST: Reset SBC6 0 = DISABLE 1 = ENABLE
8	RW	ENABLE	SWR_SBC5_RST: Reset SBC5 0 = DISABLE 1 = ENABLE
7	RW	ENABLE	SWR_I2C4_RST: Reset I2C4 0 = DISABLE 1 = ENABLE
6	RW	ENABLE	SWR_I2S4_RST: Reset I2S4 0 = DISABLE 1 = ENABLE
5	RW	ENABLE	SWR_I2S3_RST: Reset I2S3 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
3	RW	ENABLE	SWR_MSELECT_RST: Reset MSELECT 0 = DISABLE 1 = ENABLE
2	RW	0x1	SWR_3D2_RST: Reserved.
1	RO	X	SWR_CPULP_RST: Reserved.
0	RO	X	SWR_CPUG_RST: Reserved.

6.9.118 CLK_RST_CONTROLLER_RST_DEVICES_W_0

Offset: 0x35c │ Read/Write: R/W │ Reset: 0xXf407fff (0b1xx11111x10xxxxxx111111111111111)

Bit	R/W	Reset	Description
31	RW	ENABLE	SWR_EMC1_RST: Reset EMC1. 0 = DISABLE 1 = ENABLE
30	RO	X	SWR_MC1_RST: Reset MC1. This bit is disabled. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE
28	RW	ENABLE	SWR_XUSB_SS_RST: Reset XUSB_SS logic. 0 = DISABLE 1 = ENABLE
27	RW	ENABLE	SWR_DVFS_RST: Reset CLDVFS 0 = DISABLE 1 = ENABLE
26	RW	ENABLE	SWR_ADX0_RST: Reset ADX0 0 = DISABLE 1 = ENABLE
25	RW	ENABLE	SWR_AMX0_RST: Reset AMX0 0 = DISABLE 1 = ENABLE
21	RW	DISABLE	SWR_ENTROPY_RST: Reset Entropy logic. 0 = DISABLE 1 = ENABLE
14	RW	ENABLE	SWR_XUSB_PADCTL_RST: Reset XUSB PADCTL logic. 0 = DISABLE 1 = ENABLE
13	RW	0x1	SWR_RESERVED10_RST: Reserved
12	RW	0x1	SWR_RESERVED9_RST: Reserved
11	RW	0x1	SWR_RESERVED8_RST: Reserved
10	RW	0x1	SWR_RESERVED7_RST: Reserved
9	RW	0x1	SWR_RESERVED6_RST: Reserved
8	RW	ENABLE	SWR_CEC_RST: Reset CEC 0 = DISABLE 1 = ENABLE
7	RW	0x1	SWR_RESERVED5_RST: Reserved
6	RW	0x1	SWR_RESERVED4_RST: Reserved
5	RW	0x1	SWR_RESERVED3_RST: Reserved
4	RW	0x1	SWR_RESERVED2_RST: Reserved

Bit	R/W	Reset	Description
3	RW	0x1	SWR_RESERVED1_RST: Reserved
2	RW	0x1	SWR_RESERVED0_RST: Reserved
1	RW	ENABLE	SWR_SATACOLD_RST: Reset SATACOLD 0 = DISABLE 1 = ENABLE
0	RW	ENABLE	SWR_HDA2HDMICODEC_RST: Reset HDA2HDMICODEC (no dedicated module or clock.). 0 = DISABLE 1 = ENABLE

6.9.119 CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0

Offset: 0x360 │ Read/Write: R/W │ Reset: 0x007e0000 (0b00000000011111100000000000000000)

Bit	Reset	Description
29	DISABLE	CLK_ENB_HDA: Enable clock to High Definition Audio 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_SATA: Enable clock to SATA 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_SATA_OOB: Enable clock to SATA_OOB 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_EXTPERIPH3: Enable clock to EXTPERIPH3 0 = DISABLE 1 = ENABLE
25	DISABLE	CLK_ENB_EXTPERIPH2: Enable clock to EXTPERIPH2 0 = DISABLE 1 = ENABLE
24	DISABLE	CLK_ENB_EXTPERIPH1: Enable clock to EXTPERIPH1 0 = DISABLE 1 = ENABLE
23	DISABLE	CLK_ENB_ACTMON: Enable clock to ACTMON 0 = DISABLE 1 = ENABLE
22	ENABLE	CLK_ENB_SPDIF_DOUBLER: Enable S/PDIF audio sync clock doubler. 0 = DISABLE 1 = ENABLE
21	ENABLE	CLK_ENB_I2S4_DOUBLER: Enable I2S4 audio sync clock doubler. 0 = DISABLE 1 = ENABLE
20	ENABLE	CLK_ENB_I2S3_DOUBLER: Enable I2S3 audio sync clock doubler. 0 = DISABLE 1 = ENABLE
19	ENABLE	CLK_ENB_I2S2_DOUBLER: Enable I2S2 audio sync clock doubler. 0 = DISABLE 1 = ENABLE
18	ENABLE	CLK_ENB_I2S1_DOUBLER: Enable I2S1 audio sync clock doubler. 0 = DISABLE 1 = ENABLE
17	ENABLE	CLK_ENB_I2S0_DOUBLER: Enable I2S0 audio sync clock doubler. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
16	DISABLE	CLK_ENB_ATOMICS: Enable clock to ATOMICS 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_HDA2CODEC_2X: Enable clock to HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_DAM2: Enable clock to DAM2 0 = DISABLE 1 = ENABLE
13	DISABLE	CLK_ENB_DAM1: Enable clock to DAM1 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_DAM0: Enable clock to DAM0 0 = DISABLE 1 = ENABLE
11	DISABLE	CLK_ENB_APBIF: Enable clock to APBIF 0 = DISABLE 1 = ENABLE
10	DISABLE	CLK_ENB_AUDIO: Enable clock to AUDIO 0 = DISABLE 1 = ENABLE
9	DISABLE	CLK_ENB_SBC6: Enable clock to SBC6 0 = DISABLE 1 = ENABLE
8	DISABLE	CLK_ENB_SBC5: Enable clock to SBC5 0 = DISABLE 1 = ENABLE
7	DISABLE	CLK_ENB_I2C4: Enable clock to I2C4 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_I2S4: Enable clock to I2S4 0 = DISABLE 1 = ENABLE
5	DISABLE	CLK_ENB_I2S3: Enable clock to I2S3 0 = DISABLE 1 = ENABLE
4	DISABLE	CLK_ENB_TSENSOR: Enable clock to TSENSOR 0 = DISABLE 1 = ENABLE
3	DISABLE	CLK_ENB_MSELECT: Enable clock to MSELECT 0 = DISABLE 1 = ENABLE
2	0x0	CLK_ENB_3D2: Reserved.
1	DISABLE	CLK_ENB_CPULP: Enable clock to CPULP. 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_CPUG: Enable clock to CPUG. 0 = DISABLE 1 = ENABLE

6.9.120 CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0

Offset: 0x364 | Read/Write: R/W | Reset: 0x002000fc (0b00000000x01000000x00000011111100)

Bit	Reset	Description
31	DISABLE	CLK_ENB_EMC1: Enable clock to EMC1 0 = DISABLE 1 = ENABLE
30	DISABLE	CLK_ENB_MC1: Enable clock to MC1 0 = DISABLE 1 = ENABLE
29	DISABLE	CLK_ENB_EMC_DLL: Enable clock to EMC DLL calibration 0 = DISABLE 1 = ENABLE
28	DISABLE	CLK_ENB_XUSB_SS: Enable clock to XUSB_SS 0 = DISABLE 1 = ENABLE
27	DISABLE	CLK_ENB_DVFS: Enable clock to CLDVFS 0 = DISABLE 1 = ENABLE
26	DISABLE	CLK_ENB_ADX0: Enable clock to ADX0 0 = DISABLE 1 = ENABLE
25	DISABLE	CLK_ENB_AMX0: Enable clock to AMX0 0 = DISABLE 1 = ENABLE
21	0x1	CLK_ENB_ENTROPY: Reserved.
20	DISABLE	CLK_ENB_DSIB_LP: Enable clock to DSIB LP. 0 = DISABLE 1 = ENABLE
19	DISABLE	CLK_ENB_DSIA_LP: Enable clock to DSIA LP. 0 = DISABLE 1 = ENABLE
18	DISABLE	CLK_ENB_CILE: Enable clock to CSI CILE. 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_CILCD: Enable clock to CSI CILC and CILD. 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_CILAB: Enable clock to CSI CILA and CILB. 0 = DISABLE 1 = ENABLE
15	DISABLE	CLK_ENB_XUSB: Enable clock to XUSB. 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_EMC1_IOBIST: Enable clock to EMC1 IOBIST. 0 = DISABLE 1 = ENABLE
13	DISABLE	CLK_ENB_MIPI_IOBIST: Enable clock to MIPI IOBIST. 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_SATA_IOBIST: Enable clock to SATA IOBIST. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	DISABLE	CLK_ENB_HDMI_IOBIST: Enable clock to HDMI IOBIST. 0 = DISABLE 1 = ENABLE
10	DISABLE	CLK_ENB_EMC_IOBIST: Enable clock to EMC IOBIST. 0 = DISABLE 1 = ENABLE
9	DISABLE	CLK_ENB_PCIE2_IOBIST: Enable clock to PCIE2 IOBIST. 0 = DISABLE 1 = ENABLE
8	DISABLE	CLK_ENB_CEC: Enable clock to CEC. 0 = DISABLE 1 = ENABLE
7	0x1	CLK_ENB_PCIERX5: Reserved.
6	0x1	CLK_ENB_PCIERX4: Reserved.
5	0x1	CLK_ENB_PCIERX3: Reserved.
4	0x1	CLK_ENB_PCIERX2: Reserved.
3	0x1	CLK_ENB_PCIERX1: Reserved.
2	0x1	CLK_ENB_PCIERX0: Reserved.
1	DISABLE	CLK_ENB_RESERVE0: Unused clock enable for satacoldrstn 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_HDA2HDMICODEC: Enable clock to HDA2HDMICODEC (no dedicated module or clock). 0 = DISABLE 1 = ENABLE

6.9.121 CLK_RST_CONTROLLER_CCLKG_BURST_POLICY_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x368 │ Read/Write: R/W │ Reset: 0x10X00000 (0b00010000xxxxxxx00000000000000000)

Bit	R/W	Reset	Description
31:28	RW	0x1	CPU_STATE: 0000=32 kHz Clock source; 0001=IDLE Clock Source; 001X=Run clock source; 01XX=IRQ Clock Source; 1XXX=FIQ Clock Source 0 = STDBY 1 = IDLE 2 = RUN 4 = IRQ 8 = FIQ
27	RW	0x0	COP_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1 = Burst on COP FIQ
26	RW	0x0	CPU_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1 = Burst on CPU FIQ
25	RW	0x0	COP_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1 = Burst on COP IRQ
24	RW	0x0	CPU_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1 = Burst on CPU IRQ
23	RO	X	TSENSOR_SLOWDOWN: 0 = Normal ; 1 = cpug_clk/2 triggered by the temperature sensor.
16	RW	0x0	CCLK_RESERVED: Reserved. cpulp uses this bit for div2 bypass.
15:12	RW	0x0	CWAKEUP_FIQ_SOURCE: 0000 = clk_m, 0001 = pllC_out0, 0010 = clk_s, 0011 = pllM_out0, 0100 = pllP_out0, 0101 = pllP_out4, 0110 = pllC2_out0, 0111 = pllC3_out0, 1000 = PLLX_out0 (low jitter), 1001 = dvfs_cpu_clk, 1110 = PLLX_out0, 1111 = dvfs_cpu_clk, (low jitter) 0 = CLKM

Bit	R/W	Reset	Description
			1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ
11:8	RW	0x0	CWAKEUP_IRQ_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ
7:4	RW	0x0	CWAKEUP_RUN_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ
3:0	RW	0x0	CWAKEUP_IDLE_SOURCE: Same definitions as CWAKEUP_FIQ 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = DVFS_CPU_CLK 14 = PLLX_OUT0 15 = DVFS_CPU_CLK_LJ

6.9.122 CLK_RST_CONTROLLER_SUPER_CCLKG_DIVIDER_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = Disable divider. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
30	DISABLE	SUPER_CDIV_USE_THERM_CONTROLS: 1 = Use thermal controls for pulse skipper 0 = DISABLE 1 = ENABLE
28	DISABLE	CCLK_INVERT_DCD: 1 = Enable inversion of DCD (duty-cycle distortion), 0 = Disable inversion of DCD 0 = DISABLE 1 = ENABLE See “Invert Duty-Cycle Distortion Control” in this section for more information.
27	0x0	SUPER_CDIV_DIS_FROM_COP_FIQ: 0 = enable FIQ, disable FIQ.
26	0x0	SUPER_CDIV_DIS_FROM_CPU_FIQ: 0 = enable FIQ, disable FIQ.
25	0x0	SUPER_CDIV_DIS_FROM_COP_IRQ: 0 = enable IRQ, disable IRQ.
24	0x0	SUPER_CDIV_DIS_FROM_CPU_IRQ: 0 = enable IRQ, disable IRQ.
23:16	0x0	CCLK_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_CDIV_DIVISOR: Actual value = n + 1.

6.9.123 CLK_RST_CONTROLLER_CCLKLP_BURST_POLICY_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

WARNING! The PLLX_DIV2_BYPASS_LP default is low selecting source (0b1000)= pllx/2.

This is not a glitch-less switch. Do not change the register field while CPULP is using PLLX as a clock source.

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

Bit	R/W	Reset	Description
31:28	RW	0x1	CPULP_STATE: 0000=32 kHz Clock source; 0001=IDLE Clock Source; 001X=Run clock source; 01XX=IRQ Clock Source; 1XXX=FIQ Clock Source 0 = STDBY 1 = IDLE 2 = RUN 4 = IRQ 8 = FIQ
27	RW	0x0	COP_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1=Burst on COP FIQ
26	RW	0x0	CPU_AUTO_CWAKEUP_FROM_FIQ: 0 = NOP ; 1=Burst on CPU FIQ
25	RW	0x0	COP_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1=Burst on COP IRQ
24	RW	0x0	CPU_AUTO_CWAKEUP_FROM_IRQ: 0 = NOP ; 1=Burst on CPU IRQ
23	RO	X	RESERVED: Reserved for tsensor_slowdown status for cpug.
16	RW	0x0	PLLX_DIV2_BYPASS_LP: 0 = PLLX/2 ; 1 = PLLX/1 for source=1000
15:12	RW	0x0	CWAKEUP_FIQ_SOURCE: 0000 = clk_m, 0001 = pllC_out0, 0010 = clk_s, 0011 = pllM_out0, 0100 = pllP_out0, 0101 = pllP_out4, 0110 = pllC2_out0, 0111 = pllC3_out0, 1000 = PLLX_out0 (low jitter), 1001 = Reserved, 1110 = PLLX_out0, 1111 = Reserved 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0_LJ 8 = PLLX_OUT0

Bit	R/W	Reset	Description
			9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15
11:8	RW	0x0	CWAKEUP_IRQ_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15
7:4	RW	0x0	CWAKEUP_RUN_SOURCE: Same definitions as CWAKEUP_FIQ_SOURCE 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15
3:0	RW	0x0	CWAKEUP_IDLE_SOURCE: Same definitions as CWAKEUP_FIQ 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLC2_OUT0 7 = PLLC3_OUT0 8 = PLLX_OUT0_LJ 9 = RESERVED9 14 = PLLX_OUT0 15 = RESERVED15

6.9.124 CLK_RST_CONTROLLER_SUPER_CCLKLP_DIVIDER_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = Disable divider. 0 = DISABLE 1 = ENABLE
30	DISABLE	RESERVED: Reserved for cpug override of thermal control 0 = DISABLE 1 = ENABLE
28	DISABLE	CCLK_INVERT_DCD: 1 = Enable inversion of DCD (duty-cycle distortion), 0 = Disable inversion of DCD 0 = DISABLE 1 = ENABLE See “Invert Duty-Cycle Distortion Control” in this section for more information.
27	0x0	SUPER_CDIV_DIS_FROM_COP_FIQ: 0 = enable FIQ, disable FIQ.

Bit	Reset	Description
26	0x0	SUPER_CDIV_DIS_FROM_CPU_FIQ: 0 = enable FIQ, disable FIQ.
25	0x0	SUPER_CDIV_DIS_FROM_COP_IRQ: 0 = enable IRQ, disable IRQ.
24	0x0	SUPER_CDIV_DIS_FROM_CPU_IRQ: 0 = enable IRQ, disable IRQ.
23:16	0x0	CCLK_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_CDIV_DIVISOR: Actual value = n + 1.

6.9.125 CLK_RST_CONTROLLER_CLK_CPUG_CMPLX_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details..

Offset: 0x378 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxxxx00)

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

6.9.126 CLK_RST_CONTROLLER_CLK_CPULP_CMPLX_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details..

Offset: 0x37c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
8	0x0	CPULP0_CLK_STP: 1 = CPULP0 clock stop, 0 = CPULP0 clock run.

6.9.127 CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL_0

Earlier Tegra chips had a single state-machine that took care of sequencing reset (delaying reset deassertion based on a timer value) in case of a flow-controller request or a WatchDog Timer (WDT) expiry request. This has changed in Tegra 4 devices because flow-controller initiated requests do more. The timer control of reset deassertion for WDT initiated requests is in the same place while the new controls have been added to the new registers.

Offset: 0x380 │ Read/Write: R/W │ Reset: 0x00000010 (0bxxxxxxxxxxxxxxxxxxxxxxxx00010000)

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

6.9.128 CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL1_0

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

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

6.9.129 CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL2_0

Offset: 0x388 | Read/Write: R/W | Reset: 0x07000200 (0bxxxx011100000000xxxx001000000000)

Bit	Reset	Description
27:16	0x700	CAR2PMC_NONCPU_ACK_WIDTH: Counter value for ack deassertion from car2pmc for c0nc/c1nc/crail. Program this field to 1700 cycles of 300M sclk. (~70 cycles of osc_clk-12M)
11:0	0x200	CAR2PMC_CPU_ACK_WIDTH: Counter value for ack deassertion from car2pmc for fcpu0/fcpu1/fcpu2/fcpu3/scpu. Program this field to 500 cycles of 300M sclk. (~20 cycles of osc_clk - 12M)

6.9.130 CLK_RST_CONTROLLER_CLK_SOURCE_G3D2_0

Offset: 0x3b0 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	PLLM_OUT0	G3D2_CLK_SRC: 000 = pllM_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = pllA_out0 0 = PLLM_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = PLLA_OUT0
15:8	0x0	G3D2_IDLE_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x). If all 0s, this idle divisor field will not be used. For non-zero values, when HOST1x is idle, this field will be used instead of G3D2_CLK_DIVISOR.
7:0	0x0	G3D2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.131 CLK_RST_CONTROLLER_CLK_SOURCE_MSELECT_0

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

Bit	Reset	Description
31:29	CLK_M	MSELECT_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	MSELECT_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.132 CLK_RST_CONTROLLER_CLK_SOURCE_TSENSOR_0

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

Bit	Reset	Description
31:29	CLK_S	TSENSOR_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = clk_m, 110 = clk_s 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = CLK_M 6 = CLK_S
7:0	0x0	TSENSOR_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.133 CLK_RST_CONTROLLER_CLK_SOURCE_I2S3_0

Offset: 0x3bc | Read/Write: R/W | Reset: 0xd0000000 (0b11x1xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	I2S3_CLK_SRC: 00 = pllA_out0, 01 = audio SYNC_CLK 1x or 2x, 10 = pllP_out0, 11 = clk_m 0 = PLLA_OUT0 1 = SYNC_CLK 2 = PLLP_OUT0 3 = CLK_M
28	0x1	I2S3_MASTER_CLKEN: Reserved.
7:0	0x0	I2S3_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.134 CLK_RST_CONTROLLER_CLK_SOURCE_I2S4_0

Offset: 0x3c0 | Read/Write: R/W | Reset: 0xd0000000 (0b11x1xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	I2S4_CLK_SRC: 00 = pllA_out0, 01 = audio SYNC_CLK 1x or 2x, 10 = pllP_out0, 11 = clk_m 0 = PLLA_OUT0 1 = SYNC_CLK 2 = PLLP_OUT0 3 = CLK_M
28	0x1	I2S4_MASTER_CLKEN: Reserved.
7:0	0x0	I2S4_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.135 CLK_RST_CONTROLLER_CLK_SOURCE_I2C4_0

Offset: 0x3c4 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:29	CLK_M	I2C4_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
15:0	0x0	I2C4_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 1.0x)

6.9.136 CLK_RST_CONTROLLER_CLK_SOURCE_SBC5_0

Offset: 0x3c8 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SBC5_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SBC5_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.137 CLK_RST_CONTROLLER_CLK_SOURCE_SBC6_0

Offset: 0x3cc | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	SBC6_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	SBC6_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.138 CLK_RST_CONTROLLER_CLK_SOURCE_AUDIO_0

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

Bit	Reset	Description
31:29	CLK_M	AUDIO_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m, 111 = audio_clk_src_alt 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M 7 = CLK_SRC_ALT
28	0x1	AUDIO_MASTER_CLKEN: Reserved.
20	ENABLE	AUDIO_CLK_SRC_DIS: 0 = Enable audio_clk_src_alt 0 = ENABLE 1 = DISABLE
19:16	SPDIFIN	AUDIO_CLK_SRC_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK
7:0	0x0	AUDIO_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.139 CLK_RST_CONTROLLER_CLK_SOURCE_DAM0_0

Offset: 0x3d8 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxx00000xxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DAM0_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m, 111 = dam0_clk_src_alt 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M 7 = CLK_SRC_ALT

Bit	Reset	Description
28	0x1	DAM0_MASTER_CLKEN: Reserved.
20	ENABLE	DAM0_CLK_SRC_DIS: 0 = Enable dam0_clk_src_alt 0 = ENABLE 1 = DISABLE
19:16	SPDIFIN	DAM0_CLK_SRC_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK
7:0	0x0	DAM0_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.140 CLK_RST_CONTROLLER_CLK_SOURCE_DAM1_0

Offset: 0x3dc │ Read/Write: R/W │ Reset: 0xd0000000 (0b1101xxxxxxx00000xxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DAM1_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m, 111 = dam1_clk_src_alt 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M 7 = CLK_SRC_ALT
28	0x1	DAM1_MASTER_CLKEN: Reserved.
20	ENABLE	DAM1_CLK_SRC_DIS: 0 = Enable dam1_clk_src_alt 0 = ENABLE 1 = DISABLE
19:16	SPDIFIN	DAM1_CLK_SRC_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK
7:0	0x0	DAM1_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.141 CLK_RST_CONTROLLER_CLK_SOURCE_DAM2_0

Offset: 0x3e0 │ Read/Write: R/W │ Reset: 0xd0000000 (0b1101xxxxxxx00000xxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DAM2_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m, 111 = dam2_clk_src_alt 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0

Bit	Reset	Description
		4 = PLLP_OUT0 6 = CLK_M 7 = CLK_SRC_ALT
28	0x1	DAM2_MASTER_CLKEN: Reserved.
20	ENABLE	DAM2_CLK_SRC_DIS: 0 = Enable dam2_clk_src_alt 0 = ENABLE 1 = DISABLE
19:16	SPDIFIN	DAM2_CLK_SRC_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK
7:0	0x0	DAM2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.142 CLK_RST_CONTROLLER_CLK_SOURCE_HDA2CODEC_2X_0

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

Bit	Reset	Description
31:29	CLK_M	HDA2CODEC_2X_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	HDA2CODEC_2X_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.143 CLK_RST_CONTROLLER_CLK_SOURCE_ACTMON_0

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

Bit	Reset	Description
31:29	CLK_M	ACTMON_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = clk_s, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = CLK_S 6 = CLK_M
7:0	0x0	ACTMON_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.144 CLK_RST_CONTROLLER_CLK_SOURCE_EXTPERIPH1_0

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

Bit	Reset	Description
31:29	CLK_M	EXTPERIPH1_CLK_SRC: 000 = pllA_out0, 001 = clk_s, 010 = pllP_out0, 011 = clk_m, 100 = pllE_out0 0 = PLLA_OUT0

Bit	Reset	Description
		1 = CLK_S 2 = PLLP_OUT0 3 = CLK_M 4 = PLLE_OUT0
7:0	0x0	EXTPERIPH1_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.145 CLK_RST_CONTROLLER_CLK_SOURCE_EXTPERIPH2_0

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

Bit	Reset	Description
31:29	CLK_M	EXTPERIPH2_CLK_SRC: 000 = pllA_out0, 001 = clk_s, 010 = pllP_out0, 011 = clk_m, 100 = pllE_out0 0 = PLLA_OUT0 1 = CLK_S 2 = PLLP_OUT0 3 = CLK_M 4 = PLLE_OUT0
7:0	0x0	EXTPERIPH2_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.146 CLK_RST_CONTROLLER_CLK_SOURCE_EXTPERIPH3_0

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

Bit	Reset	Description
31:29	CLK_M	EXTPERIPH3_CLK_SRC: 000 = pllA_out0, 001 = clk_s, 010 = pllP_out0, 011 = clk_m, 100 = pllE_out0 0 = PLLA_OUT0 1 = CLK_S 2 = PLLP_OUT0 3 = CLK_M 4 = PLLE_OUT0
7:0	0x0	EXTPERIPH3_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.147 CLK_RST_CONTROLLER_CLK_SOURCE_NAND_SPEED_0

Offset: 0x3f8 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	NAND_SPEED_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	NAND_SPEED_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.148 CLK_RST_CONTROLLER_CLK_SOURCE_I2C_SLOW_0

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

Bit	Reset	Description
31:29	CLK_M	I2C_SLOW_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = clk_s, 110 = clk_m 0 = PLLP_OUT0

Bit	Reset	Description
		1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = CLK_S 6 = CLK_M
7:0	0x0	I2C_SLOW_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.149 CLK_RST_CONTROLLER_CLK_SOURCE_SYS_0

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

Offset: 0x400 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SYS_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.150 CLK_RST_CONTROLLER_CLK_SOURCE_SATA_OOB_0

Offset: 0x420 | Read/Write: R/W | Reset: 0xc0000000 (0b11xxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	SATA_OOB_CLK_SRC: 00 = pllP_out0, 01 = pllC_out0, 10 = pllM_out0, 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
7:0	0x0	SATA_OOB_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.151 CLK_RST_CONTROLLER_CLK_SOURCE_SATA_0

Offset: 0x424 | Read/Write: R/W | Reset: 0xc1000000 (0b11xxxxx1xxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	SATA_CLK_SRC: 00 = pllP_out0, 01 = pllC_out0, 10 = pllM_out0, 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
24	ENABLE	SATA_AUX_CLK_ENB: ENABLE SATA Tx/Rx clocks 0 = DISABLE 1 = ENABLE
7:0	0x0	SATA_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.152 CLK_RST_CONTROLLER_CLK_SOURCE_HDA_0

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

Bit	Reset	Description
31:29	CLK_M	HDA_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M

Bit	Reset	Description
7:0	0x0	HDA_CLK_DIVISOR: N = Divide by (n+1) (lsb denotes 0.5x)

6.9.153 CLK_RST_CONTROLLER_RST_DEV_V_SET_0

Offset: 0x430 | Read/Write: R/W | Reset: 0xff81ffeX (0b111111111xxxxxx111111111111x11xx)

Bit	R/W	Reset	Description
29	RW	0x1	SET_HDA_RST: Set the reset for the HDA 0 = DISABLE 1 = ENABLE
28	RW	0x1	SET_SATA_RST: Set the reset for the SATA 0 = DISABLE 1 = ENABLE
23	RW	0x1	SET_ACTMON_RST: Set the reset for the ACTMON 0 = DISABLE 1 = ENABLE
16	RW	0x1	SET_ATOMICS_RST: Set the reset for the ATOMICS 0 = DISABLE 1 = ENABLE
15	RW	0x1	SET_HDA2CODEC_2X_RST: Set the reset for the HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	RW	0x1	SET_DAM2_RST: Set the reset for the DAM2 0 = DISABLE 1 = ENABLE
13	RW	0x1	SET_DAM1_RST: Set the reset for the DAM1 0 = DISABLE 1 = ENABLE
12	RW	0x1	SET_DAM0_RST: Set the reset for the DAM0 0 = DISABLE 1 = ENABLE
11	RW	0x1	SET_APBIF_RST: Set the reset for the APBIF 0 = DISABLE 1 = ENABLE
10	RW	0x1	SET_AUDIO_RST: Set the reset for the AUDIO 0 = DISABLE 1 = ENABLE
9	RW	0x1	SET_SBC6_RST: Set the reset for the SBC6 0 = DISABLE 1 = ENABLE
8	RW	0x1	SET_SBC5_RST: Set the reset for the SBC5 0 = DISABLE 1 = ENABLE
7	RW	0x1	SET_I2C4_RST: Set the reset for the I2C4 0 = DISABLE 1 = ENABLE
6	RW	0x1	SET_I2S4_RST: Set the reset for the I2S4 0 = DISABLE 1 = ENABLE
5	RW	0x1	SET_I2S3_RST: Set the reset for the I2S3 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
3	RW	0x1	SET_MSELECT_RST: Set the reset for the MSELECT 0 = DISABLE 1 = ENABLE
2	RW	0x1	SET_3D2_RST: Set the reset for the 3D2 controller. 0 = DISABLE 1 = ENABLE
1	RO	X	SET_CPULP_RST: Set the reset for the CPULP. [CCLK Multi-Address]. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE
0	RO	X	SET_CPUG_RST: Set the reset for the CPUG. [CCLK Multi-Address]. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

6.9.154 CLK_RST_CONTROLLER_RST_DEV_V_CLR_0

Offset: 0x434 │ Read/Write: R/W │ Reset: 0xff81ffeX (0b111111111xxxxxx111111111111x11xx)

Bit	R/W	Reset	Description
29	RW	0x1	CLR_HDA_RST: Clear the reset for HDA 0 = DISABLE 1 = ENABLE
28	RW	0x1	CLR_SATA_RST: Clear the reset for SATA 0 = DISABLE 1 = ENABLE
23	RW	0x1	CLR_ACTMON_RST: Clear the reset for ACTMON 0 = DISABLE 1 = ENABLE
16	RW	0x1	CLR_ATOMICS_RST: Clear the reset for ATOMICS 0 = DISABLE 1 = ENABLE
15	RW	0x1	CLR_HDA2CODEC_2X_RST: Clear the reset for HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	RW	0x1	CLR_DAM2_RST: Clear the reset for DAM2 0 = DISABLE 1 = ENABLE
13	RW	0x1	CLR_DAM1_RST: Clear the reset for DAM1 0 = DISABLE 1 = ENABLE
12	RW	0x1	CLR_DAM0_RST: Clear the reset for DAM0 0 = DISABLE 1 = ENABLE
11	RW	0x1	CLR_APBIF_RST: Clear the reset for APBIF 0 = DISABLE 1 = ENABLE
10	RW	0x1	CLR_AUDIO_RST: Clear the reset for AUDIO 0 = DISABLE 1 = ENABLE
9	RW	0x1	CLR_SBC6_RST: Clear the reset for SBC6 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
8	RW	0x1	CLR_SBC5_RST: Clear the reset for SBC5 0 = DISABLE 1 = ENABLE
7	RW	0x1	CLR_I2C4_RST: Clear the reset for I2C4 0 = DISABLE 1 = ENABLE
6	RW	0x1	CLR_I2S4_RST: Clear the reset for I2S4 0 = DISABLE 1 = ENABLE
5	RW	0x1	CLR_I2S3_RST: Clear the reset for I2S3 0 = DISABLE 1 = ENABLE
3	RW	0x1	CLR_MSELECT_RST: Clear the reset for MSELECT 0 = DISABLE 1 = ENABLE
2	RW	0x1	CLR_3D2_RST: Clear the reset for 3D2 controller. 0 = DISABLE 1 = ENABLE
1	RO	X	CLR_CPULP_RST: Clear the reset for CPULP. [CCLK Multi-Address]. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE
0	RO	X	CLR_CPUG_RST: Clear the reset for CPUG. [CCLK Multi-Address]. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE

6.9.155 CLK_RST_CONTROLLER_RST_DEV_W_SET_0

Offset: 0x438 | Read/Write: R/W | Reset: 0xXf407fff (0b1xx11111x10xxxxxx111111111111111)

Bit	R/W	Reset	Description
31	RW	0x1	SET_EMC1_RST: Set the reset for the EMC1 0 = DISABLE 1 = ENABLE
30	RO	X	SET_MC1_RST: Set the reset for the MC1. This bit is disabled. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE
28	RW	0x1	SET_XUSB_SS_RST: Set the reset for the XUSB_SS logic 0 = DISABLE 1 = ENABLE
27	RW	0x1	SET_DVFS_RST: Set the reset for the CLDVFS 0 = DISABLE 1 = ENABLE
26	RW	0x1	SET_ADX0_RST: Set the reset for the ADX0 0 = DISABLE 1 = ENABLE
25	RW	0x1	SET_AMX0_RST: Set the reset for the AMX0 0 = DISABLE 1 = ENABLE
21	RW	0x0	SET_ENTROPY_RST: Set the reset for the ENTROPY logic. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
14	RW	0x1	SET_XUSB_PADCTL_RST: Set the reset for the XUSB_PADCTL logic. 0 = DISABLE 1 = ENABLE
13	RW	0x1	SET_RESERVED10_RST: Reserved.
12	RW	0x1	SET_RESERVED9_RST: Reserved.
11	RW	0x1	SET_RESERVED8_RST: Reserved.
10	RW	0x1	SET_RESERVED7_RST: Reserved.
9	RW	0x1	SET_RESERVED6_RST: Reserved.
8	RW	0x1	SET_CEC_RST: Set the reset for the CEC 0 = DISABLE 1 = ENABLE
7	RW	0x1	SET_RESERVED5_RST: Reserved.
6	RW	0x1	SET_RESERVED4_RST: Reserved.
5	RW	0x1	SET_RESERVED3_RST: Reserved.
4	RW	0x1	SET_RESERVED2_RST: Reserved.
3	RW	0x1	SET_RESERVED1_RST: Reserved.
2	RW	0x1	SET_RESERVED0_RST: Reserved.
1	RW	0x1	SET_SATACOLD_RST: Set the reset for the SATACOLD 0 = DISABLE 1 = ENABLE
0	RW	0x1	SET_HDA2HDMICODEC_RST: Set the reset for the HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

6.9.156 CLK_RST_CONTROLLER_RST_DEV_W_CLR_0

Offset: 0x43c │ Read/Write: R/W │ Reset: 0xXf407fff (0b1xx11111x10xxxxxx111111111111111)

Bit	R/W	Reset	Description
31	RW	0x1	CLR_EMC1_RST: Clear the reset for EMC1 0 = DISABLE 1 = ENABLE
30	RO	X	CLR_MC1_RST: Clear the reset for MC1. This bit is disabled. This bit can be written but it always reads as 0. 0 = DISABLE 1 = ENABLE
28	RW	0x1	CLR_XUSB_SS_RST: Clear the reset for the XUSB_SS logic. 0 = DISABLE 1 = ENABLE
27	RW	0x1	CLR_DVFS_RST: Clear the reset for CLDVFS 0 = DISABLE 1 = ENABLE
26	RW	0x1	CLR_ADX0_RST: Clear the reset for ADX0 0 = DISABLE 1 = ENABLE
25	RW	0x1	CLR_AMX0_RST: Clear the reset for AMX0 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
21	RW	0x0	CLR_ENTROPY_RST: Clear the reset for the ENTROPY logic 0 = DISABLE 1 = ENABLE
14	RW	0x1	CLR_XUSB_PADCTL_RST: Clear the reset for XUSB_PADCTL logic 0 = DISABLE 1 = ENABLE
13	RW	0x1	CLR_RESERVED10_RST: Reserved.
12	RW	0x1	CLR_RESERVED9_RST: Reserved.
11	RW	0x1	CLR_RESERVED8_RST: Reserved.
10	RW	0x1	CLR_RESERVED7_RST: Reserved.
9	RW	0x1	CLR_RESERVED6_RST: Reserved.
8	RW	0x1	CLR_CEC_RST: Clear the reset for CEC 0 = DISABLE 1 = ENABLE
7	RW	0x1	CLR_RESERVED5_RST: Reserved.
6	RW	0x1	CLR_RESERVED4_RST: Reserved.
5	RW	0x1	CLR_RESERVED3_RST: Reserved.
4	RW	0x1	CLR_RESERVED2_RST: Reserved.
3	RW	0x1	CLR_RESERVED1_RST: Reserved.
2	RW	0x1	CLR_RESERVED0_RST: Reserved.
1	RW	0x1	CLR_SATACOLD_RST: Clear the reset for SATACOLD 0 = DISABLE 1 = ENABLE
0	RW	0x1	CLR_HDA2HDMICODEC_RST: Clear the reset for HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

6.9.157 CLK_RST_CONTROLLER_CLK_ENB_V_SET_0

Offset: 0x440 │ Read/Write: R/W│ Reset: 0x007e0000 (0b00000000011111100000000000000000)

Bit	Reset	Description
29	0x0	SET_CLK_ENB_HDA: Set the enable clock to HDA 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_SATA: Set the enable clock to SATA 0 = DISABLE 1 = ENABLE
27	0x0	SET_CLK_ENB_SATA_OOB: Set the enable clock to SATA_OOB 0 = DISABLE 1 = ENABLE
26	0x0	SET_CLK_ENB_EXTPERIPH3: Set the enable clock to EXTPERIPH3 0 = DISABLE 1 = ENABLE
25	0x0	SET_CLK_ENB_EXTPERIPH2: Set the enable clock to EXTPERIPH2 0 = DISABLE 1 = ENABLE
24	0x0	SET_CLK_ENB_EXTPERIPH1: Set the enable clock to EXTPERIPH1 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	0x0	SET_CLK_ENB_ACTMON: Set the enable clock to ACTMON 0 = DISABLE 1 = ENABLE
22	0x1	SET_CLK_ENB_SPDIF_DOUBLER: Set the enable clock to S/PDIF audio sync doubler 0 = DISABLE 1 = ENABLE
21	0x1	SET_CLK_ENB_I2S4_DOUBLER: Set the enable clock to I2S4 audio sync doubler 0 = DISABLE 1 = ENABLE
20	0x1	SET_CLK_ENB_I2S3_DOUBLER: Set the enable clock to I2S3 audio sync doubler 0 = DISABLE 1 = ENABLE
19	0x1	SET_CLK_ENB_I2S2_DOUBLER: Set the enable clock to I2S2 audio sync doubler 0 = DISABLE 1 = ENABLE
18	0x1	SET_CLK_ENB_I2S1_DOUBLER: Set the enable clock to I2S1 audio sync doubler 0 = DISABLE 1 = ENABLE
17	0x1	SET_CLK_ENB_I2S0_DOUBLER: Set the enable clock to I2S0 audio sync doubler 0 = DISABLE 1 = ENABLE
16	0x0	SET_CLK_ENB_ATOMICS: Set the enable clock to ATOMICS 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_HDA2CODEC_2X: Set the enable clock to HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	0x0	SET_CLK_ENB_DAM2: Set the enable clock to DAM2 0 = DISABLE 1 = ENABLE
13	0x0	SET_CLK_ENB_DAM1: Set the enable clock to DAM1 0 = DISABLE 1 = ENABLE
12	0x0	SET_CLK_ENB_DAM0: Set the enable clock to DAM0 0 = DISABLE 1 = ENABLE
11	0x0	SET_CLK_ENB_APBIF: Set the enable clock to APBIF 0 = DISABLE 1 = ENABLE
10	0x0	SET_CLK_ENB_AUDIO: Set the enable clock to AUDIO 0 = DISABLE 1 = ENABLE
9	0x0	SET_CLK_ENB_SBC6: Set the enable clock to SBC6 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_SBC5: Set the enable clock to SBC5 0 = DISABLE 1 = ENABLE
7	0x0	SET_CLK_ENB_I2C4: Set the enable clock to I2C4 0 = DISABLE 1 = ENABLE
6	0x0	SET_CLK_ENB_I2S4: Set the enable clock to I2S4 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	SET_CLK_ENB_I2S3: Set the enable clock to I2S3 0 = DISABLE 1 = ENABLE
4	0x0	SET_CLK_ENB_TSENSOR: Set the enable clock to TSENSOR 0 = DISABLE 1 = ENABLE
3	0x0	SET_CLK_ENB_MSELECT: Set the enable clock to MSELECT 0 = DISABLE 1 = ENABLE
2	0x0	SET_CLK_ENB_3D2: Set the enable clock to 3D2 controller. 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_CPULP: Set the enable clock to CPULP. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE
0	0x0	SET_CLK_ENB_CPUG: Set the enable clock to CPUG. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE

6.9.158 CLK_RST_CONTROLLER_CLK_ENB_V_CLR_0

Offset: 0x444 | Read/Write: R/W | Reset: 0x007e0000 (0b00000000011111100000000000000000)

Bit	Reset	Description
29	0x0	CLR_CLK_ENB_HDA: Clear the enable clock to HDA 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_SATA: Clear the enable clock to SATA 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_SATA_OOB: Clear the enable clock to SATA_OOB 0 = DISABLE 1 = ENABLE
26	0x0	CLR_CLK_ENB_EXTPERIPH3: Clear the enable clock to EXTPERIPH3 0 = DISABLE 1 = ENABLE
25	0x0	CLR_CLK_ENB_EXTPERIPH2: Clear the enable clock to EXTPERIPH2 0 = DISABLE 1 = ENABLE
24	0x0	CLR_CLK_ENB_EXTPERIPH1: Clear the enable clock to EXTPERIPH1 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CLK_ENB_ACTMON: Clear the enable clock to ACTMON 0 = DISABLE 1 = ENABLE
22	0x1	CLR_CLK_ENB_SPDIF_DOUBLER: Clear the enable clock to S/PDIF audio sync doubler 0 = DISABLE 1 = ENABLE
21	0x1	CLR_CLK_ENB_I2S4_DOUBLER: Clear the enable clock to I2S4 audio sync doubler 0 = DISABLE 1 = ENABLE
20	0x1	CLR_CLK_ENB_I2S3_DOUBLER: Clear the enable clock to I2S3 audio sync doubler 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19	0x1	CLR_CLK_ENB_I2S2_DOUBLER: Clear the enable clock to I2S2 audio sync doubler 0 = DISABLE 1 = ENABLE
18	0x1	CLR_CLK_ENB_I2S1_DOUBLER: Clear the enable clock to I2S1 audio sync doubler 0 = DISABLE 1 = ENABLE
17	0x1	CLR_CLK_ENB_I2S0_DOUBLER: Clear the enable clock to I2S0 audio sync doubler 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CLK_ENB_ATOMICS: Clear the enable clock to ATOMICS 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_HDA2CODEC_2X: Clear the enable clock to HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	0x0	CLR_CLK_ENB_DAM2: Clear the enable clock to DAM2 0 = DISABLE 1 = ENABLE
13	0x0	CLR_CLK_ENB_DAM1: Clear the enable clock to DAM1 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_DAM0: Clear the enable clock to DAM0 0 = DISABLE 1 = ENABLE
11	0x0	CLR_CLK_ENB_APBIF: Clear the enable clock to APBIF 0 = DISABLE 1 = ENABLE
10	0x0	CLR_CLK_ENB_AUDIO: Clear the enable clock to AUDIO 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CLK_ENB_SBC6: Clear the enable clock to SBC6 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CLK_ENB_SBC5: Clear the enable clock to SBC5 0 = DISABLE 1 = ENABLE
7	0x0	CLR_CLK_ENB_I2C4: Clear the enable clock to I2C4 0 = DISABLE 1 = ENABLE
6	0x0	CLR_CLK_ENB_I2S4: Clear the enable clock to I2S4 0 = DISABLE 1 = ENABLE
5	0x0	CLR_CLK_ENB_I2S3: Clear the enable clock to I2S3 0 = DISABLE 1 = ENABLE
4	0x0	CLR_CLK_ENB_TSENSOR: Clear the enable clock to TSENSOR 0 = DISABLE 1 = ENABLE
3	0x0	CLR_CLK_ENB_MSELECT: Clear the enable clock to MSELECT 0 = DISABLE 1 = ENABLE
2	0x0	CLR_CLK_ENB_3D2: Clear the enable clock to 3D2 controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	CLR_CLK_ENB_CPULP: Clear the enable clock to CPULP. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CLK_ENB_CPUG: Clear the enable clock to CPUG. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE

6.9.159 CLK_RST_CONTROLLER_CLK_ENB_W_SET_0

Offset: 0x448 | Read/Write: R/W | Reset: 0x002000fc (0b00000000x01000000000000011111100)

Bit	Reset	Description
31	0x0	SET_CLK_ENB_EMC1: Set the enable clock to EMC1 0 = DISABLE 1 = ENABLE
30	0x0	SET_CLK_ENB_MC1: Set the enable clock to MC1 0 = DISABLE 1 = ENABLE
29	0x0	SET_CLK_ENB_EMC_DLL: Set the enable clock to EMC_DLL 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_XUSB_SS: Set the enable clock to XUSB_SS 0 = DISABLE 1 = ENABLE
27	0x0	SET_CLK_ENB_DVFS: Set the enable clock to CLDVFS 0 = DISABLE 1 = ENABLE
26	0x0	SET_CLK_ENB_ADX0: Set the enable clock to ADX0 0 = DISABLE 1 = ENABLE
25	0x0	SET_CLK_ENB_AMX0: Set the enable clock to AMX0 0 = DISABLE 1 = ENABLE
21	0x1	SET_CLK_ENB_ENTROPY: Set the enable clock to ENTROPY. 0 = DISABLE 1 = ENABLE
20	0x0	SET_CLK_ENB_DSIB_LP: Set the enable clock to DSIB LP. 0 = DISABLE 1 = ENABLE
19	0x0	SET_CLK_ENB_DSIA_LP: Set the enable clock to DSIA LP. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CLK_ENB_CILE: Set the enable clock to CSI CILE. 0 = DISABLE 1 = ENABLE
17	0x0	SET_CLK_ENB_CILCD: Set the enable clock to CSI CILC and CILD. 0 = DISABLE 1 = ENABLE
16	0x0	SET_CLK_ENB_CILAB: Set the enable clock to CSI CILA and CILB. 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_XUSB: Set the enable clock to XUSB 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
14	0x0	SET_CLK_ENB_EMC1_IOBIST: Set the enable clock to EMC1 IOBIST 0 = DISABLE 1 = ENABLE
13	0x0	SET_CLK_ENB_MIPI_IOBIST: Set the enable clock to MIPI IOBIST 0 = DISABLE 1 = ENABLE
12	0x0	SET_CLK_ENB_SATA_IOBIST: Set the enable clock to SATA IOBIST 0 = DISABLE 1 = ENABLE
11	0x0	SET_CLK_ENB_HDMI_IOBIST: Set the enable clock to HDMI IOBIST 0 = DISABLE 1 = ENABLE
10	0x0	SET_CLK_ENB_EMC_IOBIST: Set the enable clock to EMC IOBIST 0 = DISABLE 1 = ENABLE
9	0x0	SET_CLK_ENB_PCIE2_IOBIST: Set the enable clock to PCIE2 IOBIST 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_CEC: Set the enable clock to CEC 0 = DISABLE 1 = ENABLE
7	0x1	SET_CLK_ENB_PCIERX5: Set the enable clock to PCIERX5 0 = DISABLE 1 = ENABLE
6	0x1	SET_CLK_ENB_PCIERX4: Set the enable clock to PCIERX4 0 = DISABLE 1 = ENABLE
5	0x1	SET_CLK_ENB_PCIERX3: Set the enable clock to PCIERX3 0 = DISABLE 1 = ENABLE
4	0x1	SET_CLK_ENB_PCIERX2: Set the enable clock to PCIERX2 0 = DISABLE 1 = ENABLE
3	0x1	SET_CLK_ENB_PCIERX1: Set the enable clock to PCIERX1 0 = DISABLE 1 = ENABLE
2	0x1	SET_CLK_ENB_PCIERX0: Set the enable clock to PCIERX0 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_RESERVED0: Reserved SATACOLD 0 = DISABLE 1 = ENABLE
0	0x0	SET_CLK_ENB_HDA2HDMICODEC: Set the enable clock to HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

6.9.160 CLK_RST_CONTROLLER_CLK_ENB_W_CLR_0

Offset: 0x44c │ Read/Write: R/W │ Reset: 0x002000fc (0b00000000x01000000000000011111100)

Bit	Reset	Description
31	0x0	CLR_CLK_ENB_EMC1: Clear the enable clock to EMC1 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
30	0x0	CLR_CLK_ENB_MC1: Clear the enable clock to MC1 0 = DISABLE 1 = ENABLE
29	0x0	CLR_CLK_ENB_EMC_DLL: Clear the enable clock to EMC_DLL 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_XUSB_SS: Clear the enable clock to XUSB_SS 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_DVFS: Clear the enable clock to CLDVFS 0 = DISABLE 1 = ENABLE
26	0x0	CLR_CLK_ENB_ADX0: Clear the enable clock to ADX0 0 = DISABLE 1 = ENABLE
25	0x0	CLR_CLK_ENB_AMX0: Clear the enable clock to AMX0 0 = DISABLE 1 = ENABLE
21	0x1	CLR_CLK_ENB_ENTROPY: Clear the enable clock to ENTROPY. 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CLK_ENB_DSIB_LP: Clear the enable clock to DSIB LP. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CLK_ENB_DSIA_LP: Clear the enable clock to DSIA LP. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_CILE: Clear the enable clock to CSI CILE. 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CLK_ENB_CILCD: Clear the enable clock to CSI CILC and CILD. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CLK_ENB_CILAB: Clear the enable clock to CSI CILA and CILB. 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_XUSB: Clear the enable clock to XUSB 0 = DISABLE 1 = ENABLE
14	0x0	CLR_CLK_ENB_EMC1_IOBIST: Clear the enable clock to EMC1 IOBIST 0 = DISABLE 1 = ENABLE
13	0x0	CLR_CLK_ENB_MIPI_IOBIST: Clear the enable clock to MIPI IOBIST 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_SATA_IOBIST: Clear the enable clock to SATA IOBIST 0 = DISABLE 1 = ENABLE
11	0x0	CLR_CLK_ENB_HDMI_IOBIST: Clear the enable clock to HDMI IOBIST 0 = DISABLE 1 = ENABLE
10	0x0	CLR_CLK_ENB_EMC_IOBIST: Clear the enable clock to EMC IOBIST 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	0x0	CLR_CLK_ENB_PCIE2_IOBIST: Clear the enable clock to PCIE2 IOBIST 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CLK_ENB_CEC: Clear the enable clock to CEC 0 = DISABLE 1 = ENABLE
7	0x1	CLR_CLK_ENB_PCIERX5: Clear the enable clock to PCIERX5 0 = DISABLE 1 = ENABLE
6	0x1	CLR_CLK_ENB_PCIERX4: Clear the enable clock to PCIERX4 0 = DISABLE 1 = ENABLE
5	0x1	CLR_CLK_ENB_PCIERX3: Clear the enable clock to PCIERX3 0 = DISABLE 1 = ENABLE
4	0x1	CLR_CLK_ENB_PCIERX2: Clear the enable clock to PCIERX2 0 = DISABLE 1 = ENABLE
3	0x1	CLR_CLK_ENB_PCIERX1: Clear the enable clock to PCIERX1 0 = DISABLE 1 = ENABLE
2	0x1	CLR_CLK_ENB_PCIERX0: Clear the enable clock to PCIERX0 0 = DISABLE 1 = ENABLE
1	0x0	CLR_CLK_ENB_RESERVED0: Reserved.
0	0x0	CLR_CLK_ENB_HDA2HDMICODEC: Clear the enable clock to HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

6.9.161 CLK_RST_CONTROLLER_RST_CPUG_CMPLX_SET_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

Bit	Reset	Description
30	0x0	SET_PRESETDBG: 1 = assert nPRESETDBG to the CoreSight. JTAG connection to an external debugger will be lost. 0 = DISABLE 1 = ENABLE
29	0x1	SET_NONCPURESET: 1 = Assert reset to the whole nonCPU region of the CPU. For FCCPLEX, by default the NONCPU region is not power-gated but the reset is asserted because CRAIL is power-gated by default. 0 = DISABLE 1 = ENABLE
28	0x0	SET_PERIPHRESET: Reserved.
24	0x0	SET_L2RESET: 1 = Assert nL2RESET to CPU. 0 = DISABLE 1 = ENABLE
23	0x0	SET_CXRESET3: 1 = Assert nCXRESET to CPU3. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	0x0	SET_CXRESET2: 1 = Assert nCXRESET to CPU2. 0 = DISABLE 1 = ENABLE
21	0x0	SET_CXRESET1: 1 = Assert nCXRESET to CPU1. 0 = DISABLE 1 = ENABLE
20	0x0	SET_CXRESET0: 1 = Assert nCXRESET to CPU0. 0 = DISABLE 1 = ENABLE
19	0x0	SET_CORERESET3: 1 = Assert nCORERESET to CPU3. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CORERESET2: 1 = Assert nCORERESET to CPU2. 0 = DISABLE 1 = ENABLE
17	0x0	SET_CORERESET1: 1 = Assert nCORERESET to CPU1. 0 = DISABLE 1 = ENABLE
16	0x0	SET_CORERESET0: 1 = Assert nCORERESET to CPU0. 0 = DISABLE 1 = ENABLE
15	0x1	SET_DBGRESET3: 1 = Assert nDBGRESET to CPU3. 0 = DISABLE 1 = ENABLE
14	0x1	SET_DBGRESET2: 1 = Assert nDBGRESET to CPU2. 0 = DISABLE 1 = ENABLE
13	0x1	SET_DBGRESET1: 1 = Assert nDBGRESET to CPU1. 0 = DISABLE 1 = ENABLE
12	0x1	SET_DBGRESET0: 1 = Assert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
11	0x1	SET_WDRESET3: Reserved.
10	0x1	SET_WDRESET2: Reserved.
9	0x1	SET_WDRESET1: Reserved.
8	0x0	SET_WDRESET0: Reserved.
7	0x1	SET_DERESET3: Reserved.
6	0x1	SET_DERESET2: Reserved.
5	0x1	SET_DERESET1: Reserved.
4	0x0	SET_DERESET0: Reserved.
3	0x1	SET_CPURESET3: 1 = Assert nCPUPORRESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	SET_CPURESET2: 1 = Assert nCPUPORRESET to CPU2. 0 = DISABLE 1 = ENABLE
1	0x1	SET_CPURESET1: 1 = Assert nCPUPORRESET to CPU1. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x1	SET_CPURESET0: 1 = Assert nCPUPORRESET to CPU0. 0 = DISABLE 1 = ENABLE

6.9.162 CLK_RST_CONTROLLER_RST_CPUG_CMPLX_CLR_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x454 │ Read/Write: R/W │ Reset: 0x2000feef (0bx010xxx00000000011111110111011111)

Bit	Reset	Description
30	0x0	CLR_PRESETDBG: 1 = Deassert nPRESETDBG to the CoreSight. 0 = DISABLE 1 = ENABLE
29	0x1	CLR_NONCPURESET: 1 = Deassert reset to the whole nonCPU region of the CPU. For FCCPLEX, by default the NONCPU region is not power-gated but the reset is asserted because CRAIL is power-gated by default. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_PERIPHRESET: Reserved.
24	0x0	CLR_L2RESET: 1 = Deassert nL2RESET to CPU. 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CXRESET3: 1 = Deassert nCXRESET to CPU3. 0 = DISABLE 1 = ENABLE
22	0x0	CLR_CXRESET2: 1 = Deassert nCXRESET to CPU2. 0 = DISABLE 1 = ENABLE
21	0x0	CLR_CXRESET1: 1 = Deassert nCXRESET to CPU1. 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CXRESET0: 1 = Deassert nCXRESET to CPU0. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CORERESET3: 1 = Deassert nCORERESET to CPU3. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CORERESET2: 1 = Deassert nCORERESET to CPU2. 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CORERESET1: 1 = Deassert nCORERESET to CPU1. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CORERESET0: 1 = Deassert nCORERESET to CPU0. 0 = DISABLE 1 = ENABLE
15	0x1	CLR_DBGRESET3: 1 = Deassert nDBGRESET to CPU3. 0 = DISABLE 1 = ENABLE
14	0x1	CLR_DBGRESET2: 1 = Deassert nDBGRESET to CPU2. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
13	0x1	CLR_DBGRESET1: 1 = Deassert nDBGRESET to CPU1. 0 = DISABLE 1 = ENABLE
12	0x1	CLR_DBGRESET0: 1 = Deassert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
11	0x1	CLR_WDRESET3: Reserved.
10	0x1	CLR_WDRESET2: Reserved.
9	0x1	CLR_WDRESET1: Reserved.
8	0x0	CLR_WDRESET0: Reserved.
7	0x1	CLR_DERESET3: Reserved.
6	0x1	CLR_DERESET2: Reserved.
5	0x1	CLR_DERESET1: Reserved.
4	0x0	CLR_DERESET0: Reserved.
3	0x1	CLR_CPURESET3: 1 = Deassert nCPUPORRESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	CLR_CPURESET2: 1 = Deassert nCPUPORRESET to CPU2. 0 = DISABLE 1 = ENABLE
1	0x1	CLR_CPURESET1: 1 = Deassert nCPUPORRESET to CPU1. 0 = DISABLE 1 = ENABLE
0	0x1	CLR_CPURESET0: 1 = Deassert nCPUPORRESET to CPU0. 0 = DISABLE 1 = ENABLE

6.9.163 CLK_RST_CONTROLLER_RST_CPULP_CMPLX_SET_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x458 │ Read/Write: R/W │ Reset: 0x2000feef (0bx010xxx00000000011111111011101111)

Bit	Reset	Description
30	0x0	SET_PRESETDBG: 1 = assert nPRESETDBG to the CoreSight. JTAG connection to an external debugger will be lost. 0 = DISABLE 1 = ENABLE
29	0x1	SET_NONCPURESET: 1 = Assert reset to the whole nonCPU region of the CPU. 0 = DISABLE 1 = ENABLE
28	0x0	SET_PERIPHRESET: Reserved
24	0x0	SET_L2RESET: 1 = Assert nL2RESET to the CPU. 0 = DISABLE 1 = ENABLE
23	0x0	SET_CXRESET3: N/A
22	0x0	SET_CXRESET2: N/A
21	0x0	SET_CXRESET1: N/A

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

6.9.164 CLK_RST_CONTROLLER_RST_CPULP_CMPLX_CLR_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x45c │ Read/Write: R/W │ Reset: 0x20001001 (0bx010xxx0xxx0xxx0xxx1xxx0xxx0xxx1)

Bit	Reset	Description
30	0x0	CLR_PRESETDBG: 1 = Deassert nPRESETDBG to the CoreSight. 0 = DISABLE 1 = ENABLE
29	0x1	CLR_NONCPURESET: 1 = Deassert reset to the whole nonCPU region of the CPU. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_PERIPHRESET: Reserved.
24	0x0	CLR_L2RESET: 1 = Deassert nL2RESET to CPU. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	CLR_CXRESET0: 1 = Deassert nCXRESET to CPU0. 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CORERESET0: 1 = Deassert nCORERESET to CPU0. 0 = DISABLE 1 = ENABLE
12	0x1	CLR_DBGRESET0: 1 = Deassert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
8	0x0	CLR_WDRESET0: Reserved.
4	0x0	CLR_DERESET0: Reserved.
0	0x1	CLR_CPURESET0: 1 = Deassert nCPUPORRESET to CPU0. 0 = DISABLE 1 = ENABLE

6.9.165 CLK_RST_CONTROLLER_CLK_CPUG_CMPLX_SET_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

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

6.9.166 CLK_RST_CONTROLLER_CLK_CPUG_CMPLX_CLR_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

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

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

6.9.167 CLK_RST_CONTROLLER_CLK_CPULP_CMPLX_SET_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x468 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
8	0x0	SET_CPU0_CLK_STP: 1 = Assert CPU0 clock stop 0 = DISABLE 1 = ENABLE

6.9.168 CLK_RST_CONTROLLER_CLK_CPULP_CMPLX_CLR_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x46c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
8	0x0	CLR_CPU0_CLK_STP: 1 = Deassert CPU0 clock stop 0 = DISABLE 1 = ENABLE

6.9.169 CLK_RST_CONTROLLER_CPU_CMPLX_STATUS_0

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

Bit	Reset	Description
30	X	PRESETDBG: ENABLE = nPRESETDBG asserted to the CoreSight. JTAG connection to external debugger will be lost. 0 = DISABLE 1 = ENABLE
29	X	NONCPURESET: ENABLE = Reset asserted to the whole nonCPU region of the CPU 0 = DISABLE 1 = ENABLE
28	X	MCRESET: ENABLE = mcreset_ asserted 0 = DISABLE 1 = ENABLE
27	X	CSITEPTMRESET: ENABLE = nCSITEPTMRESET asserted 0 = DISABLE 1 = ENABLE
26	X	AXICIFRESET: ENABLE = nAXICIFRESET asserted 0 = DISABLE 1 = ENABLE
25	X	MPSELECTRESET: ENABLE = mselectreset asserted 0 = DISABLE 1 = ENABLE
24	X	L2RESET: ENABLE = L2RESET asserted to the CPU 0 = DISABLE 1 = ENABLE
23	X	CXRESET3: ENABLE = nCXRESET asserted to CPU3 0 = DISABLE 1 = ENABLE
22	X	CXRESET2: ENABLE = nCXRESET asserted to CPU2 0 = DISABLE 1 = ENABLE
21	X	CXRESET1: ENABLE = nCXRESET asserted to CPU1 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	X	CXRESET0: ENABLE = nCXRESET asserted to CPU0 0 = DISABLE 1 = ENABLE
19	X	CORERESET3: ENABLE = nCORERESET asserted to CPU3 0 = DISABLE 1 = ENABLE
18	X	CORERESET2: ENABLE = nCORERESET asserted to CPU2 0 = DISABLE 1 = ENABLE
17	X	CORERESET1: ENABLE = nCORERESET asserted to CPU1 0 = DISABLE 1 = ENABLE
16	X	CORERESET0: ENABLE = nCORERESET asserted to CPU0 0 = DISABLE 1 = ENABLE
15	X	DBGRESET3: ENABLE = nDBGRESET asserted to CPU3 0 = DISABLE 1 = ENABLE
14	X	DBGRESET2: ENABLE = nDBGRESET asserted to CPU2 0 = DISABLE 1 = ENABLE
13	X	DBGRESET1: ENABLE = nDBGRESET asserted to CPU1 0 = DISABLE 1 = ENABLE
12	X	DBGRESET0: ENABLE = nDBGRESET asserted to CPU0 0 = DISABLE 1 = ENABLE
11	X	WDRESET3: Reserved.
10	X	WDRESET2: Reserved.
9	X	WDRESET1: Reserved.
8	X	WDRESET0: Reserved.
3	X	CPURESET3: ENABLE = nCPUPORRESET asserted to CPU3 0 = DISABLE 1 = ENABLE
2	X	CPURESET2: ENABLE = nCPUPORRESET asserted to CPU2 0 = DISABLE 1 = ENABLE
1	X	CPURESET1: ENABLE = nCPUPORRESET asserted to CPU1 0 = DISABLE 1 = ENABLE
0	X	CPURESET0: ENABLE = nCPUPORRESET asserted to CPU0 0 = DISABLE 1 = ENABLE

6.9.170 CLK_RST_CONTROLLER_INTSTATUS_0

Interrupt Status Register.

- car_int[0] = axicifreset
- car_int[1] = l2reset
- car_int[2] = cpuporreset_cpu0
- car_int[3] = cpuporreset_cpu1

- car_int[4] = cpuporreset_cpu2
- car_int[5] = cpuporreset_cpu3
- car_int[6] = corereset_cpu0
- car_int[7] = corereset_cpu1
- car_int[8] = corereset_cpu2
- car_int[9] = corereset_cpu3
- car_int[10] = tsensor2car_slowdown_sclk
- car_int[11] = spare
- car_int[12] = spare
- car_int[13] = spare
- car_int[14] = spare
- car_int[15] = spare
- car_int[16] = spare
- car_int[17] = spare
- car_int[18] = spare
- car_int[19] = spare

Offset: 0x478 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

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

6.9.171 CLK_RST_CONTROLLER_INTMASK_0

Interrupt Mask Register.

Offset: 0x47c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

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

6.9.172 CLK_RST_CONTROLLER_UTMIP_PLL_CFG0_0

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

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

UTMIP PLL Configuration Register 0

Offset: 0x480 │ Read/Write: R/W │ Reset: 0x00500100 (0bx00000001010000000000001xxxxx000)

Bit	Reset	Description
30:29	0x0	UTMIP_PLL_VREG_BG_CTRL: Voltage regulator and band-gap probe mux control. VREG_BG_CTRL = 0 => VREG_BG_PRB = 0. VREG_BG_CTRL = 1 => VREG_BG_PRB = bandgap. VREG_BG_CTRL = 2 => VREG_BG_PRB = Vregulator. See cell specification.
28:27	0x0	UTMIP_PLL_VREG_CTRL: Voltage regulator voltage level control. See cell specification.

Bit	Reset	Description
26:25	0x0	UTMIP_PLL_KCP: KCP of the UTMIP PHY PLL. Charge Pump Gain control. Default value is zero. See cell specification.
23:16	0x50	UTMIP_PLL_NDIV: NDIV[7:0] input of UTMIP PLL. This is the feedback divider on the VCO feedback. 0x0 is not allowed. See cell specification.
15:8	0x1	UTMIP_PLL_MDIV: MDIV[7:0] input of the UTMIP PLL. This is the predivide on the PLL. 0x0 is not allowed. See cell specification.
2	0x0	UTMIP_PLL_LOCK_OVR: LOCK_OVERRIDE control of the UTMIP PLL. Forces PLL_LOCK to 1. See cell specification.
1	0x0	UTMIP_PLL_EN_FSTLCK: EN_FSTLCK input of UTMIP PLL. Reserving pin for PLL fast lock circuitry. See cell specification.
0	0x0	UTMIP_PLL_ENB_LCKDET: ENB_LCKDET input of UTMIP PLL. When set to 1, powers down the lock detect. Setting ENB_LCKDET=X, LOCK_OVERRIDE=1, PLL_LOCK=1, and PLL_FREQLOCK=1 also powers down lock detect. 0 0 0->1(after lock) 0->1 (after lock) 1 0 0 0 lock detect power down. See cell specification.

6.9.173 CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0

UTMIP PLL Configuration and Parameters

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

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

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

- PllUOnState: start pllu_enable_count and pll_lock_count
  - Wait ~1 µs to enable the PLL_U (pllu_enable_count == ClkXtal * PLLU_ENABLE_DLY_COUNT * 8)
  - Wait ~1 ms until PLL_U is stable (pll_lock_count == ClkXtal * PLLU_STABLE_COUNT * 256) => USB_PHY PLL_ENABLE
  - pll_active_count = 0
    - Wait 5 µs to enable pll_active: pll_active_count == ClkXtal * PLL_ACTIVE_DLY_COUNT * 16 => USB_PHY PLL_ACTIVE
  - Wait ~2.5 ms until USB_PHY is stable (pll_lock_count == ClkXtal * XTAL_FREQ_COUNT * 256) => USB_PHY

Sample values for a 12 MHz Xtal clock are:

- PLLU_ENABLE_DLY_COUNT[4:0] = 1 * 12 / 8 = 1.5 = 0x02
- PLLU_STABLE_COUNT[11:0] = 1000 * 12 / 256 = 46.9 = 0x2f
- PLL_ACTIVE_DLY_COUNT[4:0] = 10 * 12 / 16 = 7.5 = 0x08
- XTAL_FREQ_COUNT[11:0] = 2500 * 12 / 256 = 117.2 = 0x76

UTMIP PLL and PLLU Configuration Register 1

Offset: 0x484 │ Read/Write: R/W │ Reset: 0x180150c0 (0b00011000000000001010100001000000)

Bit	Reset	Description
31:27	0x3	UTMIP_PLLU_ENABLE_DLY_COUNT: Controls the wait time to enable PLL_U when coming out of suspend or reset.
26:18	0x0	UTMIP_PLL_RSVD: Reserved
17	0x0	UTMIP_FORCE_PLLU_POWERUP: Force PLL_U into power up.

Bit	Reset	Description
16	0x1	UTMIP_FORCE_PLLU_POWERDOWN: Force PLL_U into power down. (Overrides FORCE_PLLU_POWERUP.)
15	0x0	UTMIP_FORCE_PLL_ENABLE_POWERUP: Force UTMIP PLL pll_enable input on.
14	0x1	UTMIP_FORCE_PLL_ENABLE_POWERDOWN: Force UTMIP PLL pll_enable input off. (Overrides FORCE_PLL_ENABLE_POWERUP.)
13	0x0	UTMIP_FORCE_PLL_ACTIVE_POWERUP: Force UTMIP PLL pll_active input on.
12	0x1	UTMIP_FORCE_PLL_ACTIVE_POWERDOWN: Force UTMIP PLL pll_active input off. (Overrides FORCE_PLL_ACTIVE_POWERUP.)
11:0	0xc0	UTMIP_XTAL_FREQ_COUNT: Determines the time to wait until the output of UTMIP PLL is considered stable.

6.9.174 CLK_RST_CONTROLLER_UTMIP_PLL_CFG2_0

UTMIP Miscellaneous Configurations

Offset: 0x488 │ Read/Write: R/W │ Reset: 0x41318015 (0bx1xxxx01001100011000000000010101)

Bit	Reset	Description
30	0x1	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
25	0x0	UTMIP_FORCE_PD_SAMP_D_POWERUP: Force UTMIP PLL PD_SAMP_D input (XUSB DEV) into power up.
24	0x1	UTMIP_FORCE_PD_SAMP_D_POWERDOWN: Force UTMIP PLL PD_SAMP_D input (XUSB DEV) into power down. (Overrides FORCE_PD_SAMP_D_POWERUP.)
23:18	0xc	UTMIP_PLL_ACTIVE_DLY_COUNT: 10 us / (1/19.2MHz) = 192 / 16 = 12
17:6	0x600	UTMIP_PLLU_STABLE_COUNT: PLLU frequency lock delay (See comments above on correct delay and calculation)
5	0x0	UTMIP_FORCE_PD_SAMP_C_POWERUP: Force UTMIP PLL PD_SAMP_C input into power up.
4	0x1	UTMIP_FORCE_PD_SAMP_C_POWERDOWN: Force UTMIP PLL PD_SAMP_C input into power down. (Overrides FORCE_PD_SAMP_C_POWERUP.)
3	0x0	UTMIP_FORCE_PD_SAMP_B_POWERUP: Force UTMIP PLL PD_SAMP_B input (XUSB HOST) into power up.
2	0x1	UTMIP_FORCE_PD_SAMP_B_POWERDOWN: Force UTMIP PLL PD_SAMP_B input (XUSB HOST) into power down. (Overrides FORCE_PD_SAMP_B_POWERUP.)
1	0x0	UTMIP_FORCE_PD_SAMP_A_POWERUP: Force UTMIP PLL PD_SAMP_A input into power up.
0	0x1	UTMIP_FORCE_PD_SAMP_A_POWERDOWN: Force UTMIP PLL PD_SAMP_A input into power down. (Overrides FORCE_PD_SAMP_A_POWERUP.)

6.9.175 CLK_RST_CONTROLLER_PLLE_AUX_0

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

Offset: 0x48c │ Read/Write: R/W │ Reset: 0x0X044070 (0b0xx0xx10000001000100000011100xx)

Bit	R/W	Reset	Description
31	RW	SKIP	PLLE_SS_SEQ_INCLUDE: 0=Skip, 1=Include. Include PLLE spread spectrum power sequencer. If this bit needs to be toggled, do it before writing '1' to PLLE_SEQ_ENABLE. 0 = SKIP 1 = INCLUDE
28	RW	0x0	PLLE_REF_SEL_PLLREFE: PLLE input reference clock source select2. 0 = Select the setting of the PLLE_REF_SRC field mentioned below, 1 = Select PLLREFE_out which is 600M (use pre-divM=50 for 12 MHz reference)
27:26	RO	X	PLLE_SEQ_STATE: PLLE power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	PLLE_SEQ_START_STATE: PLLE power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	PLLE_SEQ_ENABLE: PLLE power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
23:16	RW	0x4	PLLE_SS_DLY: PT6: Delay between spread spectrum ON<->OFF transitions in SS power toggle sequence: ON->OFF->ON. Range is 0-255 ms in 1 ms steps.
15:8	RW	0x40	PLLE_LOCK_DLY: PT5: Delay from PLLE ENABLE to lock. Range is 0-128 µs in 1 µs steps
7	RW	0x0	TEST_FAST_PT: 0=Normal timer steps. 1=Fast timer steps. Fast programmable timer steps size for testing. Normal is per µs or ms, fast is per oscillator clock. Applies to all programmable timer counters in SATA, PCIe and PLLE seq.
6	RW	0x1	PLLE_SS_SWCTL: 0=PLLE spread spectrum configuration setup by hardware, 1=Setup by software during training
5	RW	0x1	PLLE_CONFIG_SWCTL: This bit should never be written to 0. Do not change the setup bits once written by software. 0=PLLE config setup by hardware, 1=PLLE setup by software during training. Affects resettable bits when PLL is off (SETUP and EXT_SETUP bits). All other config bits are left untouched as initialized, for example, M/N/P, SS coefficient.
4	RW	0x1	PLLE_ENABLE_SWCTL: 0=PLLE enable by hardware, 1=PLLE enable by software
3	RW	0x0	PLLE_USE_LOCKDET: 0=Use programmable delay, 1=use lockdet signals from PLLE
2	RW	OSC_DIV	PLLE_REF_SRC: PLLE input reference clock source select. 0=OSC_DIV (options: mux (PLLs, ext_osc), /2, /4), 1=PLLP_OUT0 (use pre-divM=18 for 12 MHz reference) 0 = OSC_DIV 1 = PLLP_OUT0

6.9.176 CLK_RST_CONTROLLER_SATA_PLL_CFG0_0

SATA Power Sequencer Input Software Control Program Guide

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

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

Offset: 0x490 │ Read/Write: R/W │ Reset: 0x0X000003 (0bxxxxxx10xxxxxxxxxxxxxxxx0000x011)

Bit	R/W	Reset	Description
27:26	RO	X	SATA_SEQ_STATE: SATA power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	SATA_SEQ_START_STATE: SATA power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	SATA_SEQ_ENABLE: SATA power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
7	RW	0x0	SATA_SEQ_PADPLL_PD_INPUT_VALUE: 0=Pad PLL is powered up, 1=Software control pad PLL powered down (PD) by setting this and SATA_SEQ_IN_SWCTL bit
6	RW	0x0	SATA_SEQ_LANE_PD_INPUT_VALUE: 0=IO PHY is powered up, 1=Software control I/O PHY powered down by setting this and SATA_SEQ_IN_SWCTL bit
5	RW	0x0	SATA_SEQ_RESET_INPUT_VALUE: 0=SATA reset deasserted, 1=Software control reset by setting this and SATA_SEQ_IN_SWCTL bit
4	RW	0x0	SATA_SEQ_IN_SWCTL: 0=Seq input by hardware, 1=Seq input by software
2	RW	0x0	SATA_PADPLL_USE_LOCKDET: 0=Use programmable delay, 1=Use lockdet signals from PLL
1	RW	0x1	SATA_PADPLL_RESET_OVERRIDE_VALUE: 0=Pad PLL is powered up, 1=Software control reset by setting this and SATA_PADPLL_RESET_SWCTL bit
0	RW	0x1	SATA_PADPLL_RESET_SWCTL: 0=Reset by hardware, 1=Reset by software

6.9.177 CLK_RST_CONTROLLER_SATA_PLL_CFG1_0

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

Offset: 0x494 │ Read/Write: R/W │ Reset: 0x20202008 (0b00100000001000000010000000001000)

Bit	Reset	Description
31:24	0x20	SATA_LANE_IDDQ2_PADPLL_RESET_DLY: PT4: Delay from placing lane out of IDDQ to PAD PLL in reset. Range is 0-200 µs.
23:16	0x20	SATA_PADPLL_IDDQ2LANE_SLUMBER_DLY: PT3: Delay from SATA PAD PLL out of IDDQ to lane placed out of IDDQ. Range is 0-64 µs.
15:8	0x20	SATA_PADPLL_PU_POST_DLY: PT2: Delay from RESET to SATA PAD PLL lock. Range is 0-64 µs.
7:0	0x8	SATA_LANE_IDDQ2_PADPLL_IDDQ_DLY: PT1: Delay from placing lane in IDDQ to PAD PLL in IDDQ. Range is 0-16 µs.

6.9.178 CLK_RST_CONTROLLER_PCIE_PLL_CFG_0

PCIe Power Sequencer Input Software Control Program Guide

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

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

Offset: 0x498 │ Read/Write: R/W │ Reset: 0x0X00000d (0bxxxxxx10xxxxxxxxxxxxxxxxxx001101)

Bit	R/W	Reset	Description
27:26	RO	X	PCIE_SEQ_STATE: PCIe power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	PCIE_SEQ_START_STATE: PCIe power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	PCIE_SEQ_ENABLE: PCIe power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
5	RW	0x0	PCIE_SEQ_RESET_INPUT_VALUE: 0=PCIe reset deasserted, 1=Software control reset by setting this and PCIE_SEQ_IN_SWCTL bit
4	RW	0x0	PCIE_SEQ_IN_SWCTL: 0=Sequencer input by hardware, 1=Sequencer input by software
3	RW	ENABLE	PCIE_XCLK_ENABLE_OVERRIDE_VALUE: Override value used only when PCIE_XCLK_ENABLE_SWCTL is set 0 = DISABLE 1 = ENABLE
2	RW	0x1	PCIE_XCLK_ENABLE_SWCTL: 0=XCLK enabled by hardware, 1=XCLK enabled by software
1	RW	0x0	PCIE_PADPLL_RESET_OVERRIDE_VALUE: 0=PCIe iophy PLL on, 1=PLL reset. Override value used only when PCIE_PADPLL_RESET_SWCTL is set
0	RW	0x1	PCIE_PADPLL_RESET_SWCTL: 0=Reset by hardware, 1=Reset by software

6.9.179 CLK_RST_CONTROLLER_PROG_AUDIO_DLY_CLK_0

- Clock doubler with 1x/2x select for: I2S0 through I2S4 and S/PDIF.
- Clock doublers with 1x/2x select use by the clock controller for each I2S0 through I2S4 and S/PDIF sync clock.
- Clock doubler enable controlled by CLK_ENB_I2S[0..4]_DOUBLE and CLK_ENB_SPDIF_DOUBLE bits in CLK_OUT_ENB_V register. CLK_ENB_*_DOUBLE bit must be enabled for either SYNC_1X_CLK or SYNC_2X_CLK to operate.
- PROG_DLY_CLK provides programmable delay for the clock doublers. An individual clock doubler is provided for each audio sync clock: I2S0..4, S/PDIF.

Audio Sync Clock Source Mux Options

SPDIFIN, I2S0, I2S1, I2S2, I2S3, I2S4, PLLA_OUT0, EXT_VIMCLK

Enable Sync Clocks

There are 3 related clock enables/selects:

- CG_1. AUDIO_SYNC_CLK_I2S0-SYNC_CLK_DIS: disable output of sync_clk mux
- CG_2. CLK_OUT_ENB_V-CLK_ENB_I2S0_DOUBLER: disable input to doubler
- SEL_3. PROG_AUDIO_DLY_CLK-I2S0_1X_SEL: Select 1x sync clock. 0=2x 1=1x

Flow of Clock Logic

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

Suggested Programming for Sync Clock Enable:

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

SYNC_CLK_DIS	CLK_ENB_I2S0_DOUBLER	I2S0_1X_SEL	SYNC_CLK
0	1	0	2x (Default)
0	1	1	1x
1	1	X	0 (off, recommended)
1	X	X	0 (off)
X	0	X	0 (off)

Offset: 0x49c │ Read/Write: R/W │ Reset: 0x00777777 (0bxx00000011101110111011101110111)

Bit	Reset	Description
29	0x0	SPDIF_1X_SEL: Select S/PDIF 1x sync_clk. 0=sync_2x_clk, 1=sync_clk
28	0x0	I2S4_1X_SEL: Select I2S4 1x sync_clk. 0=sync_2x_clk, 1=sync_clk
27	0x0	I2S3_1X_SEL: Select I2S3 1x sync_clk. 0=sync_2x_clk, 1=sync_clk
26	0x0	I2S2_1X_SEL: Select I2S2 1x sync_clk. 0=sync_2x_clk, 1=sync_clk
25	0x0	I2S1_1X_SEL: Select I2S1 1x sync_clk. 0=sync_2x_clk, 1=sync_clk
24	0x0	I2S0_1X_SEL: Select I2S0 1x sync_clk. 0=sync_2x_clk, 1=sync_clk
23:20	0x7	SYNC_CLK_SPDIF_DELCLK_SEL: 16 taps of selectable delay for SYNC_CLK clock doubler
19:16	0x7	SYNC_CLK_I2S4_DELCLK_SEL: 16 taps of selectable delay for SYNC_CLK clock doubler
15:12	0x7	SYNC_CLK_I2S3_DELCLK_SEL: 16 taps of selectable delay for SYNC_CLK clock doubler
11:8	0x7	SYNC_CLK_I2S2_DELCLK_SEL: 16 taps of selectable delay for SYNC_CLK clock doubler
7:4	0x7	SYNC_CLK_I2S1_DELCLK_SEL: 16 taps of selectable delay for SYNC_CLK clock doubler
3:0	0x7	SYNC_CLK_I2S0_DELCLK_SEL: 16 taps of selectable delay for SYNC_CLK clock doubler

6.9.180 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S0_0

Audio Sync Clock Source Select

Offset: 0x4a0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock.
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0

Bit	Reset	Description
		7 = EXT_VIMCLK

6.9.181 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S1_0

Audio Sync Clock Source Select

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

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = reserved 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK

6.9.182 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S2_0

Audio Sync Clock Source Select

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

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK

6.9.183 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S3_0

Audio Sync Clock Source Select

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

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3

Bit	Reset	Description
		5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK

6.9.184 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S4_0

Audio Sync Clock Source Select

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

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK

6.9.185 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_SPDIF_0

Audio Sync Clock Source Select

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

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable Audio sync clock
3:0	0x0	SYNC_CLK_RATE: 0000 = SPDIFIN recovered bit clock. 0001 = I2S0 bit clock. 0010 = I2S1 bit clock. 0011 = I2S2 bit clock. 0100 = I2S3 bit clock. 0101 = I2S4 bit clock. 0110 = pllA_out0. 0111 = external vimclk (vimclk). 1xxx = Reserved. 0 = SPDIFIN 1 = I2S0 2 = I2S1 3 = I2S2 4 = I2S3 5 = I2S4 6 = PLLA_OUT0 7 = EXT_VIMCLK

6.9.186 CLK_RST_CONTROLLER_PLLD2_BASE_0

Offset: 0x4b8 | Read/Write: R/W | Reset: 0x0X00010c (0b000xx010x000xx0000000001xxx01100)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLD2_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLD2_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLD2_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE

Bit	R/W	Reset	Description
27	RO	X	PLLD2_LOCK: 0 = not lock, 1 = lock.
26	RW	0x0	PLLD2_CLKENABLE_CSI: 0 = disable, 1 = normal operation. Enable CSI fast and slow p/n clocks to pad macros.
25	RW	PLL_D2	DSIB_CLK_SRC: 0 = PLL_D 1 = PLL_D2
24	RW	0x0	CSI_CLK_SRC: 0 = Brick, 1 = PLL_D/D2.
22:20	RW	0x0	PLLD2_DIVP: 0 = Post divider (2^n).
17:8	RW	0x1	PLLD2_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLD2_DIVM: PLL input divider.

6.9.187 CLK_RST_CONTROLLER_PLLD2_MISC_0

Offset: 0x4bc │ Read/Write: R/W │ Reset: 0x00000100 (0b00000000000000000000000100000000)

Bit	Reset	Description
31	0x0	PLLD2_FO_MODE: 1 = 5-125 MHz, 0 = 40-1000 MHz.
30	0x0	PLLD2_CLKENABLE: 0 = Disable, 1 = Normal operation.
29:27	0x0	PLLD2_PTS: Base PLLD test output select. 0 = DISABLE 1 = DUTYCYCLECTL 2 = VCO 3 = NDIV 4= FO 5 = SLOWCLOCK 6 = FASTCLOCK 7 = LOADPULSE
26:24	0x0	PLLD2_LOADADJ: load pulse position adjust.
23	0x0	PLLD2_DIV_RST: 0 = normal operation, 1 = reset.
22	0x0	PLLD2_LOCK_ENABLE: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
21:16	0x0	PLLD2_LOCK_SEL: lock select.
15:12	0x0	PLLD2_DCCON: Base PLLD DCCON control.
11:8	0x1	PLLD2_CPCON: Base PLLD charge pump setup control.
7:4	0x0	PLLD2_LFCON: Base PLLD loop filter setup control.
3:0	0x0	PLLD2_VCOCON: Base PLLD VCO range setup control.

6.9.188 CLK_RST_CONTROLLER_UTMIP_PLL_CFG3_0

UTMIP_PLL_CFG3_0

Offset: 0x4c0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:0	0x0	UTMIP_PLL_SETUP: Debug control bits. Default is 0. [6:0]: Lock Detect Controls [8:7]: Current source control

Bit	Reset	Description
		[10:9]: 0 = Default is zero 1 = Force loop filter to vdda/2 2 = Force loop filter to vdda/2 with PFD OFF 3 = Force loop filter to vdda with PFD OFF [11]: Corevdd detection override [12]: VREG_CTRL [2] [15:13] Bgap_tc[2:0] [16] Bgap kickstart [17] Add regulator load/bleeder [18] Power-down regulator (when IDDQ=0) [19] Power-down isource (when IDDQ=0) [20] CLKOUTMUX60 0 = CLKOUTMUX60= 1b0 1 = CLKOUTMUX60= vcoclk/16 [23:21]: Phase selection on CLKOUTMUX60

6.9.189 CLK_RST_CONTROLLER_PLLREFE_BASE_0

Offset: 0x4c4 | Read/Write: R/W | Reset: 0x00000000 (0bx00000xxxxxx00000000000000000000)

Bit	Reset	Description
30	DISABLE	PLLREFE_ENABLE: 0 = DISABLE 1 = ENABLE
29	REF_ENABLE	PLLREFE_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. 0 = REF_ENABLE 1 = REF_DISABLE
28:27	0x0	PLLREFE_KCP
26	0x0	PLLREFE_KVCO
19:16	0x0	PLLREFE_PLDIV
15:8	0x0	PLLREFE_NDIV
7:0	0x0	PLLREFE_MDIV

6.9.190 CLK_RST_CONTROLLER_PLLREFE_MISC_0

Offset: 0x4c8 | Read/Write: R/W | Reset: 0x0X010000 (0b000xxxxxxxxxx0010000000000000000)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLREFE_EN_FSTLCK: 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLREFE_LOCK_ENABLE: 0 = DISABLE 1 = ENABLE
29	RW	0x0	PLLREFE_LOCK_OVERRIDE. Forces PLLREFE_LOCK and PLLREFE_FREQLOCK to 1.
25	RO	X	PLLREFE_FREQLOCK: 0 = frequency acquisition not achieved, 1 = frequency acquisition occurred. PLLREFE_LOCK_ENABLE must be enabled.
24	RO	X	PLLREFE_LOCK: 0 = not lock (phase+frequency), 1 = lock (phase+frequency). PLLREFE_LOCK_ENABLE must be enabled.

Bit	R/W	Reset	Description
18:17	RW	0x0	PLLREFE_PTS: 00 = PTO is 0 (DISABLE); 01 = PTO is FO; 10 = PTO is VCO out; 11 = Reserved.
16	RW	0x1	PLLREFE_IDDQ: 0 = The PLLREFE is powered up. 1 = Software can put the PLLREFE in IDDQ by setting this bit. 0 = OFF 1 = ON
15:0	RW	0x0	PLLREFE_SETUP

CPU Trimmer Registers

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

These registers control the delay through the programmable trimmers (CORE_CLOCKS_shaper) inside the Cortex™-A15 partitions.

- CPU_FINETRIM_BYP
- CPU_FINETRIM_SELECT
- CPU_FINETRIM_DR
- CPU_FINETRIM_DF
- CPU_FINETRIM_F
- CPU_FINETRIM_R

The naming convention of the fields of these registers is:

- FCPU_n : partition n in fast CPU cluster
- SCPU_n : partition n in slow CPU cluster

The partitions related to these registers are as follows:

- FCPU_1: fcpu0 partition
- FCPU_2: fcpu1 partition
- FCPU_3: fcpu2 partition
- FCPU_4: fcpu3 partition
- FCPU_5: fl2 partition
- FCPU_6: ftop partition
- SCPU_7: not used

- SCPU_8: not used
- SCPU_9: not used

Offset: 0x4d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	SCPU_9
7	0x0	SCPU_8
6	0x0	SCPU_7
5	0x0	FCPU_6
4	0x0	FCPU_5
3	0x0	FCPU_4
2	0x0	FCPU_3
1	0x0	FCPU_2
0	0x0	FCPU_1

6.9.191 CLK_RST_CONTROLLER_CPU_FINETRIM_SELECT_0

Offset: 0x4d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	SCPU_9
7	0x0	SCPU_8
6	0x0	SCPU_7
5	0x0	FCPU_6
4	0x0	FCPU_5
3	0x0	FCPU_4
2	0x0	FCPU_3
1	0x0	FCPU_2
0	0x0	FCPU_1

6.9.192 CLK_RST_CONTROLLER_CPU_FINETRIM_DR_0

Offset: 0x4d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	SCPU_9
7	0x0	SCPU_8
6	0x0	SCPU_7
5	0x0	FCPU_6
4	0x0	FCPU_5
3	0x0	FCPU_4
2	0x0	FCPU_3
1	0x0	FCPU_2
0	0x0	FCPU_1

6.9.193 CLK_RST_CONTROLLER_CPU_FINETRIM_DF_0

Offset: 0x4dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	SCPU_9
7	0x0	SCPU_8
6	0x0	SCPU_7
5	0x0	FCPU_6
4	0x0	FCPU_5
3	0x0	FCPU_4
2	0x0	FCPU_3
1	0x0	FCPU_2
0	0x0	FCPU_1

6.9.194 CLK_RST_CONTROLLER_CPU_FINETRIM_F_0

Offset: 0x4e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx000000000000000000)

Bit	Reset	Description
17:16	0x0	SCPU_9
15:14	0x0	SCPU_8
13:12	0x0	SCPU_7
11:10	0x0	FCPU_6
9:8	0x0	FCPU_5
7:6	0x0	FCPU_4
5:4	0x0	FCPU_3
3:2	0x0	FCPU_2
1:0	0x0	FCPU_1

6.9.195 CLK_RST_CONTROLLER_CPU_FINETRIM_R_0

Offset: 0x4e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx000000000000000000)

Bit	Reset	Description
17:16	0x0	SCPU_9
15:14	0x0	SCPU_8
13:12	0x0	SCPU_7
11:10	0x0	FCPU_6
9:8	0x0	FCPU_5
7:6	0x0	FCPU_4
5:4	0x0	FCPU_3
3:2	0x0	FCPU_2
1:0	0x0	FCPU_1

6.9.196 CLK_RST_CONTROLLER_PLLC2_BASE_0

There are special startup and programming considerations for this digital PLL. Consult the datasheet for details

Some field descriptions will have a LATCHED or a DIRECT keyword:

- LATCHED: The writes to this field are latched by the PLLC2 STROBE. Once one or more of the "LATCHED" fields are written, PLLC2_STROBE must be toggled by software for the value to be latched into the PLL and become effective. See 'Dynamic Frequency Change' section in the datasheet. If the sequence is not completed correctly, readback of value by software may not be the value used by PLL.
- DIRECT: It is a DIRECT read/write field. The writes to this field are not latched by STROBE. So, readback will always tell the correct state.

The digital PLL, PLL12G_DIG_B1, has an update rate limit of 19.2 MHz. The osc_div_clk that drives the CLOCKIN pin of these PLLs can have values of 12 MHz, 13 MHz, 19.2 MHz, 16.8 MHz, or 26 MHz. The 26 MHz crystal is faster than the allowed update rate. This can be resolved by programming the SDIV of the PLLs to internally divide the clock by 2, giving 13 MHz.

Offset: 0x4e8 │ Read/Write: R/W │ Reset: 0x0X000101 (0b000xxxxxx000xxxx00000001xxxxxx01)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC2_BYPASS: DIRECT. 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLC2_ENABLE: DIRECT. 0 = disable, 1 = enable. Will invert and connect to IDDQ 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLC2_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. To go into the minimum power mode for the PLL, this bit must be asserted. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLC2_FREQ_LOCK: 0 = not lock, 1 = frequency is locked.
26	RO	X	PLLC2_PHASE_LOCK: 0 = not lock, 1 = phase is locked.
22:20	RW	0x0	PLLC2_DIVP: DIRECT. 0 = post divider (divide by N+1). **PLLC2_DIVP** **PDIV** 0 1 1 2 2 3 3 4 4 6 5 8 6 12 7 16
15:8	RW	0x1	PLLC2_DIVN: PLL feedback divider. Must perform 'Dynamic Frequency Change' as noted in the data sheet for this to be changed once the PLL is running LATCHED.
1:0	RW	0x1	PLLC2_DIVM: PLL input divider. DIRECT. 00 = Divide by 1. 01 = Divide by 1. 10 = Divide by 2. 11 = Divide by 4.

6.9.197 CLK_RST_CONTROLLER_PLLC2_MISC_0_0

Offset: 0x4ec │ Read/Write: R/W │ Reset: 0x40000000 (0b01000000000000000000000000000000)

Bit	Reset	Description
31	0x0	PLLC2_STROBE: STROBE for various PLL input controls. Those controls are transparent only when STROBE is high.

Bit	Reset	Description
30	0x1	PLLC2_RESET: Reset for digital logic of the PLL. DIRECT. 0 = Out of reset, 1 = Reset asserted. 0 = DISABLE 1 = ENABLE
29:28	0x0	PLLC2_SDM_DIV: SDM Clock Divide Ratio. LATCHED. 00 = Divide by 4. 01 = Divide by 8. 10 = Divide by 16. 11 = Divide by 32.
27:26	0x0	PLLC2_FILT_DIV: Filter Clock Divide Ratio. LATCHED. 00 = Divide by 8. 01 = Divide by 16. 10 = Divide by 32. 11 = Divide by 64.
25:18	0x0	PLLC2_ALPHA: IIR pole location. LATCHED.
17:9	0x0	PLLC2_KB: IIR filter feed-forward gain. LATCHED.
8:2	0x0	PLLC2_KA: IIR filter input gain. LATCHED.
1:0	0x0	PLLC2_KVCO: VCO gain adjustment. DIRECT.

6.9.198 CLK_RST_CONTROLLER_PLLC2_MISC_1_0

Offset: 0x4f0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx000xx000000xx00xx00xxxx0000)

Bit	Reset	Description
26:24	0x0	PLLC2_VCO_BAND_SW: If calibration is not being done, this field can force the VCO band (select n+1 band, 1 to 8). LATCHED.
21:16	0x0	PLLC2_DAC_CODE_SW: For calibration at Vmin, the voltage is controlled by this field. LATCHED.
13:12	0x0	PLLC2_CAL_NUM_FRM: Number of frames during calibration for which the delta is collected for taking an average. LATCHED. 00 = 2, 01 = 4, 10 = 8, 11 = 16.
9	0x0	PLLC2_CAL_EN: Calibration is enabled (1) or disabled (0). DIRECT.
8	0x0	PLLC2_PD_AFTER_FD: Hold Phase-Detector output at '0' until Frequency Detector is locked. LATCHED. 1: Enable, 0: Disable
3:0	0x0	PLLC2_SDM_GAIN: SDM Gain Control. LATCHED.

6.9.199 CLK_RST_CONTROLLER_PLLC2_MISC_2_0

Offset: 0x4f4 | Read/Write: R/W | Reset: 0x00000000 (0bxx00xx00xx00xx00xx00xx00xx00xx00)

Bit	Reset	Description
29:28	0x0	PLLC2_PD_ULCK_FRM: pd_unlock_num_frames. Refer to the NVIDIA Tegra 4 Series 4-PLUS-1 Quad-Core Processors Data Sheet (DS-06217-001) for details. LATCHED.
25:24	0x0	PLLC2_PD_LCK_FRM: pd_lock_num_frames. Refer to the Tegra 4 data sheet for details. LATCHED.
21:20	0x0	PLLC2_PD_OUT_HYST: pd_pullout_hyst_val. Refer to the Tegra 4 data sheet for details. LATCHED.
17:16	0x0	PLLC2_PD_IN_HYST: pd_pullin_hyst_val. Refer to the Tegra 4 data sheet for details. LATCHED.
13:12	0x0	PLLC2_FD_ULCK_FRM: fd_unlock_num_frames. Refer to the Tegra 4 data sheet for details. LATCHED
9:8	0x0	PLLC2_FD_LCK_FRM: fd_lock_num_frames. Refer to the Tegra 4 data sheet for details. LATCHED
5:4	0x0	PLLC2_FD_OUT_HYST: fd_pullout_hyst_val. Refer to the Tegra 4 data sheet for details. LATCHED.

Bit	Reset	Description
1:0	0x0	PLLC2_FD_IN_HYST: fd_pullin_hyst_val. Refer to the Tegra 4 data sheet for details. LATCHED.

6.9.200 CLK_RST_CONTROLLER_PLLC2_MISC_3_0

Offset: 0x4f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	PLLC2_SETUP: Read the PLL data sheet for the description of each bit. DIRECT.

6.9.201 CLK_RST_CONTROLLER_PLLC3_BASE_0

Offset: 0x4fc | Read/Write: R/W | Reset: 0x0X000101 (0b000xxxxxx000xxxx00000001xxxxxx01)

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC3_BYPASS: DIRECT. 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLC3_ENABLE: DIRECT. 0 = disable, 1 = enable. Will invert and connect to IDDQ. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLC3_REF_DIS: 0 = enable reference clock, 1 = disable reference clock. To enter the minimum power mode for the PLL, this bit must be asserted. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLC3_FREQ_LOCK: 0 = not locked, 1 = frequency is locked.
26	RO	X	PLLC3_PHASE_LOCK: 0 = not locked, 1 = phase is locked.
22:20	RW	0x0	PLLC3_DIVP: 0 = post divider (divide by N+1). DIRECT. **PLLC3_DIVP** **PDIV** 0 1 1 2 2 3 3 4 4 6 5 8 6 12 7 16
15:8	RW	0x1	PLLC3_DIVN: PLL feedback divider. Must perform 'Dynamic Frequency Change' as noted in the data sheet for this to be changed once the PLL is running. LATCHED.
1:0	RW	0x1	PLLC3_DIVM: PLL input divider. DIRECT. 00 = Divide by 1, 01 = Divide by 1. 10 = Divide by 2. 11 = Divide by 4.

6.9.202 CLK_RST_CONTROLLER_PLLC3_MISC_0_0

Offset: 0x500 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	PLLC3_STROBE: STROBE for various PLL input controls. Those controls are transparent only when STROBE is high.
30	0x1	PLLC3_RESET: Reset for digital logic of the PLL. DIRECT. 0 = Out of reset, 1 = Reset asserted. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
29:28	0x0	PLLC3_SDM_DIV: SDM Clock Divide Ratio. LATCHED. 00 = Divide by 4. 01 = Divide by 8. 10 = Divide by 16. 11 = Divide by 32.
27:26	0x0	PLLC3_FILT_DIV: Filter Clock Divide Ratio. LATCHED. 00 = Divide by 8. 01 = Divide by 16. 10 = Divide by 32. 11 = Divide by 64.
25:18	0x0	PLLC3_ALPHA: IIR pole location. LATCHED.
17:9	0x0	PLLC3_KB: IIR filter feed-forward gain. LATCHED.
8:2	0x0	PLLC3_KA: IIR filter input gain. LATCHED.
1:0	0x0	PLLC3_KVCO: VCO gain adjustment. DIRECT.

6.9.203 CLK_RST_CONTROLLER_PLLC3_MISC_1_0

Offset: 0x504 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx000xx000000xx00xx00xxxx0000)

Bit	Reset	Description
26:24	0x0	PLLC3_VCO_BAND_SW: If calibration is not being done, this field can force the VCO band (select n+1 band, 1 to 8). LATCHED.
21:16	0x0	PLLC3_DAC_CODE_SW: For calibration at Vmin, the voltage is controlled by this field. LATCHED.
13:12	0x0	PLLC3_CAL_NUM_FRM: Number of frames during calibration for which the delta is collected for taking an average. LATCHED. 00 = 2, 01 = 4, 10 = 8, 11 = 16.
9	0x0	PLLC3_CAL_EN: Calibration is enabled (1) or disabled (0). DIRECT.
8	0x0	PLLC3_PD_AFTER_FD: Hold Phase-Detector output at '0' until Frequency Detector is locked. 1: Enable, 0: Disable. LATCHED.
3:0	0x0	PLLC3_SDM_GAIN: SDM Gain Control. LATCHED.

6.9.204 CLK_RST_CONTROLLER_PLLC3_MISC_2_0

Offset: 0x508 | Read/Write: R/W | Reset: 0x00000000 (0bxx00xx00xx00xx00xx00xx00xx00xx00)

Bit	Reset	Description
29:28	0x0	PLLC3_PD_ULCK_FRM: pd_unlock_num_frames. Refer to the NVIDIA Tegra 4 Series 4-PLUS-1 Quad-Core Processors Data Sheet (DS-06217-001) for details. LATCHED.
25:24	0x0	PLLC3_PD_LCK_FRM: pd_lock_num_frames. Refer to the Tegra 4 data sheet for details. LATCHED.
21:20	0x0	PLLC3_PD_OUT_HYST: pd_pullout_hyst_val. Refer to the Tegra 4 data sheet for details. LATCHED.
17:16	0x0	PLLC3_PD_IN_HYST: pd_pullin_hyst_val. Refer to the Tegra 4 data sheet for details. LATCHED.
13:12	0x0	PLLC3_FD_ULCK_FRM: fd_unlock_num_frames. Refer to the Tegra 4 data sheet for details. LATCHED.
9:8	0x0	PLLC3_FD_LCK_FRM: fd_lock_num_frames. Refer to the Tegra 4 data sheet for details. LATCHED.
5:4	0x0	PLLC3_FD_OUT_HYST: fd_pullout_hyst_val. Refer to the Tegra 4 data sheet for details. LATCHED.
1:0	0x0	PLLC3_FD_IN_HYST: fd_pullin_hyst_val. Refer to the Tegra 4 data sheet for details. LATCHED.

6.9.205 CLK_RST_CONTROLLER_PLLC3_MISC_3_0

Offset: 0x50c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxx00000000000000000000000000)

Bit	Reset	Description
25:0	0x0	PLLC3_SETUP: Refer to the PLL datasheet for the descriptions of each bit. DIRECT.

6.9.206 CLK_RST_CONTROLLER_PLLX_MISC_1_0

Offset: 0x510 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000000)

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

6.9.207 CLK_RST_CONTROLLER_PLLX_MISC_2_0

Offset: 0x514 │ Read/Write: R/W │ Reset: 0x0000000X (0b00000000000000000000000xx00xx00)

Bit	R/W	Reset	Description
31:24	RW	0x0	PLLX_DYNRAMP_STEPB: MISC_2 PLLX DYNRAMP
23:16	RW	0x0	PLLX_DYNRAMP_STEPA: MISC_2 PLLX DYNRAMP
15:8	RW	0x0	PLLX_NDIV_NEW: MISC_2 PLLX DYNRAMP
5	RW	0x0	PLLX_EN_FSTLCK: MISC_2 PLLX DYNRAMP
4	RW	0x0	PLLX_LOCK_OVERRIDE: MISC_2 PLLX DYNRAMP
3	RO	X	PLLX_PLL_FREQLOCK: MISC_2 PLLX DYNRAMP
2	RO	X	PLLX_DYNRAMP_DONE: MISC_2 PLLX DYNRAMP
1	RW	0x0	PLLX_CLAMP_NDIV: MISC_2 PLLX DYNRAMP
0	RW	0x0	PLLX_EN_DYNRAMP: MISC_2 PLLX DYNRAMP

6.9.208 CLK_RST_CONTROLLER_PLLX_MISC_3_0

Offset: 0x518 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000xxxx0000xxxx0000)

Bit	Reset	Description
23	0x0	SODEV_READ_ORDERING: SODEV reads ordering control. 0: Allow only one A15 SODEV read transaction at a time per CPU. 1: Allow multiple A15 SODEV read transactions at a time per CPU.
22	0x0	SODEV_WRITE_ORDERING: SODEV writes ordering control. 0: Allow only one A15 SODEV write transaction at a time per CPU. 1: Allow multiple A15 SODEV write transactions at a time per CPU.
21:19	0x0	Reserved.
18:16	0x0	PLLX_PRB_OBS_SEL: 0 = CSITE 1 = FCPU0_LEAF 2 = FCPU1_LEAF 3 = FCPU2_LEAF 4 = FCPU3_LEAF 5 = MC 6 = MSELECT 7 = FTOP_SHAPER
11:10	0x0	PLLX_PRB_DRV_CTRL:
9:8	0x0	PLLX_SEL_PRB:

Bit	Reset	Description
3	0x0	PLLX_IDDQ:0: The PLLX is powered up.1:Software can put the PLLX in IDDQ by setting this bit.
2:1	0x0	PLLX_KCP: Charge Pump Gain control
0	0x0	PLLX_KVCO:VCO gain

6.9.209 CLK_RST_CONTROLLER_XUSBIO_PLL_CFG0_0

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

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

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

Offset: 0x51c │ Read/Write: R/W │ Reset: 0x0X00000d (0bxxxxxx10xxxxxxxxxxxxxxxxx0001101)

Bit	R/W	Reset	Description
27:26	RO	X	XUSBIO_SEQ_STATE: XUSBIO power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	XUSBIO_SEQ_START_STATE: XUSBIO power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	XUSBIO_SEQ_ENABLE: XUSBIO power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
6	RW	0x0	XUSBIO_PADPLL_USE_LOCKDET: 0=Use programmable delay, 1=use lockdet signals from PLL
5	RW	0x0	XUSBIO_SEQ_RESET_INPUT_VALUE: 0=XUSBIO PLL ON, 1=XUSB IOPLL OFF. Software controls the state machine by setting this and the XUSBIO_SEQ_IN_SWCTL bit
4	RW	0x0	XUSBIO_SEQ_IN_SWCTL: 0=seq input by hardware, 1=seq input by software.
3	RW	ENABLE	XUSBIO_CLK_ENABLE_OVERRIDE_VALUE: Override value used only when XUSBIO_CLK_ENABLE_SWCTL is set 0 = DISABLE 1 = ENABLE
2	RW	0x1	XUSBIO_CLK_ENABLE_SWCTL: 0=ioclks enabled by hardware, 1=ioclks enabled by software
1	RW	0x0	XUSBIO_PADPLL_RESET_OVERRIDE_VALUE: 0=XUSBIO PLL on, 1=PLL reset. Override value used only when XUSBIO_PADPLL_RESET_SWCTL is set
0	RW	0x1	XUSBIO_PADPLL_RESET_SWCTL: 0=Reset by hardware, 1=Reset by software

6.9.210 CLK_RST_CONTROLLER_XUSBIO_PLL_CFG1_0

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

Offset: 0x520 │ Read/Write: R/W │ Reset: 0x000a0a0f (0bxxxxxxxx00001010000010100000111)

Bit	Reset	Description
23:16	0xa	XUSBIO_PADPLL_RESET2_PADPLL_IDDQ_DLY: PT1: Delay from XUSBIO PAD PLL RESET assertion to the PAD PLL IDDQ assertion. Range is 0 - 10 µs.

Bit	Reset	Description
15:8	0xa	XUSBIO_PADPLL_IDDQ2_PADPLL_RESET_DLY: PT2: Delay from XUSBIO PAD PLL IDDQ deassertion to the PAD PLL reset deassertion. Range is 0 - 10 µs.
7:0	0xf	XUSBIO_PADPLL_PU_POST_DLY: PT3: Delay from XUSBIO PAD PLL Reset deassertion to the PLL LOCK. Range is 0 -15 µs.

6.9.211 CLK_RST_CONTROLLER_PLLE_AUX1_0

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

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

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

6.9.212 CLK_RST_CONTROLLER_PLLP_RESHIFT_0

Offset: 0x528 │ Read/Write: R/W │ Reset: 0x0000003b (0bxxxxxxxxxxxxxxxxxxxxxxxx0000111011)

Bit	Reset	Description
9:2	0xe	PLLP_OUT0_RATIO: PLLP_OUT0 divider from base PLLP (lsb denotes 0.5x). The default is 408/8=51 MHz.
1	ENABLE	PLLP_OUT0_CLKEN: PLLP_OUT0 divider clock enable. 0 = DISABLE 1 = ENABLE
0	RESET_DISABLE	PLLP_OUT0_RSTN: PLLP_OUT0 divider reset. 0 = reset, 1 = not reset. 0 = RESET_ENABLE 1 = RESET_DISABLE

6.9.213 CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0

Offset: 0x52c │ Read/Write: R/W │ Reset: 0x0X00000f (0bxxxxxx10xxxxxxxxxxxxxxxx00001111)

Bit	R/W	Reset	Description
27:26	RO	X	UTMIPLL_SEQ_STATE: UTMIPLL power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	UTMIPLL_SEQ_START_STATE: UTMIPLL power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	UTMIPLL_SEQ_ENABLE:UTMIPLL power sequencer enable. Start state is loaded on first cycle after set. 0 = DISABLE 1 = ENABLE
7	RW	0x0	UTMIPLL_IDDQ_PD_INCLUDE: Include UTMIPLL IDDQ sequencing as part of PLL powerdown/powerup sequence controlled by XUSB hardware.
6	RW	0x0	UTMIPLL_USE_LOCKDET: 0=Use programmable delay, 1=use lockdet signals from PLL and programmable delay on top of that.

Bit	R/W	Reset	Description
5	RW	0x0	UTMIPLL_SEQ_RESET_INPUT_VALUE: 0=UTMIPLL ON, 1=UTMIPLL OFF. Software controls the state machine by setting this and UTMIPLL_SEQ_IN_SWCTL bit
4	RW	0x0	UTMIPLL_SEQ_IN_SWCTL: 0=seq input by hardware, 1=seq input by software
3	RW	ENABLE	UTMIPLL_CLK_ENABLE_OVERRIDE_VALUE: Override value used only when UTMIPLL_CLK_ENABLE_SWCTL is set 0 = DISABLE 1 = ENABLE
2	RW	0x1	UTMIPLL_CLK_ENABLE_SWCTL: 0=UTMIP clocks enable by hardware, 1=UTMIP clocks enable by software
1	RW	0x1	UTMIPLL_IDDQ_OVERRIDE_VALUE: 0=PLL not in IDDQ mode, 1=PLL in IDDQ mode. Override value used only when UTMIPLL_IDDQ_SWCTL is set
0	RW	0x1	UTMIPLL_IDDQ_SWCTL: 0=IDDQ by hardware, 1=IDDQ by software.

6.9.214 CLK_RST_CONTROLLER_PLLU_HW_PWRDN_CFG0_0

Offset: 0x530 │ Read/Write: R/W │ Reset: 0x0X00008d (0bxxxxxx10xxxxxxxxxxxxxxxx100011x1)

Bit	R/W	Reset	Description
27:26	RO	X	PLLU_SEQ_STATE: PLLU power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	PLLU_SEQ_START_STATE: PLLU power sequencer start state 0 = SEQ_OFF 1 = SEQ_ON
24	RW	DISABLE	PLLU_SEQ_ENABLE: PLLU power sequencer enable. Start state is loaded on the first cycle after set. 0 = DISABLE 1 = ENABLE
7	RW	0x1	PLLU_USE_SWITCH_DETECT: 0=Use programmable delay, 1=Use hardware switch detect logic.
6	RW	0x0	PLLU_USE_LOCKDET: 0=Use programmable delay, 1=Use lockdet signals from PLL and programmable delay on top of that.
5	RW	0x0	PLLU_SEQ_RESET_INPUT_VALUE: 0=PLLU ON, 1=PLLU OFF. Software controls the state machine by setting this and the PLLU_SEQ_IN_SWCTL bit
4	RW	0x0	PLLU_SEQ_IN_SWCTL: 0=Seq input by hardware, 1=Seq input by software
3	RW	ENABLE	PLLU_CLK_ENABLE_OVERRIDE_VALUE: Override value used only when PLLU_CLK_ENABLE_SWCTL is set 0 = DISABLE 1 = ENABLE
2	RW	0x1	PLLU_CLK_ENABLE_SWCTL: 0=PLLU clocks enabled by hardware, 1=PLLU clocks enabled by software
0	RW	0x1	PLLU_CLK_SWITCH_SWCTL: 0=Control SS/FS clock frequency through hardware, 1=Control FS/SS clock frequency through software

6.9.215 CLK_RST_CONTROLLER_XUSB_PLL_CFG0_0

Note that default value is the recommended range.

Offset: 0x534 | Read/Write: R/W | Reset: 0x80191480 (0b10000000000110010001010010000000)

Bit	Reset	Description
31:24	0x80	PLLU_CLK_SWITCH_DLY: Delay from SS Clock source change to the actual frequency change to 32 kHz clock. Range is ~100 µs.
23:14	0x64	PLLU_LOCK_DLY: Delay from PLLU ENABLE assertion to the PLL LOCK. Range is 100 µs-1 ms.
13:10	0x5	UTMIPLL_IDDQ2_ENABLE_DLY: Delay from UTMIPLL IDDQ deassertion to the UTMIPLL ENABLE assertion. Range ~5 µs.
9:0	0x80	UTMIPLL_LOCK_DLY: Delay from UTMIPLL ENABLE assertion to the PLL LOCK. Range 5-15 µs.

6.9.216 CLK_RST_CONTROLLER_CLK_CPU_MISC_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x53c | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx010)

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

6.9.217 CLK_RST_CONTROLLER_CLK_CPUG_MISC_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x540 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx010)

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

6.9.218 CLK_RST_CONTROLLER_CLK_CPULP_MISC_0

[CCLK Multi-Address]: See “Multi-Address Tagging” in this section for more details.

Offset: 0x544 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx010)

Bit	Reset	Description
2:0	0x2	CPULP_ATCLK_RATIO: Clock divider ratio for the ATB clocks in CCPLEX 000 = div-by-2. 001 = div-by-3. 010 = div-by-4.

Bit	Reset	Description
		011 = div-by-5. 100 = div-by-6. Other values = Reserved

6.9.219 CLK_RST_CONTROLLER_PLLX_HW_CTRL_CFG_0

Offset: 0x548 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000001)

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

6.9.220 CLK_RST_CONTROLLER_PLLX_SW_RAMP_CFG_0

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

Bit	Reset	Description
31:24	0x0	DYNRAMP_STEPA: PLL ramp rate
23:16	0x0	DYNRAMP_STEPB: PLL ramp rate
15:8	0x0	NDIV_NEW: PLL ramp rate
7:0	0x0	NDIV: PLL ramp rate

6.9.221 CLK_RST_CONTROLLER_PLLX_HW_CTRL_STATUS_0

Offset: 0x550 | Read/Write: RO | Reset: 0xXX000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	FSM_STATE: 0 = DISABLE 1 = SEQ_AWAIT 2 = SEQ_FAST_BEGIN 3 = SEQ_FAST_BUSY0 4 = SEQ_FAST_BUSY1 5 = SEQ_FAST_BUSY2 6 = SEQ_FAST_BUSY3 7 = SEQ_SLOW_BEGIN 8 = SEQ_SLOW_BUSY0

Bit	Reset	Description
		9 = SEQ_SLOW_BUSY1 10 = SEQ_SLOW_BUSY2 11 = SEQ_SLOW_BUSY3 12 = SEQ_DONE 13 = ILLEGAL
27:26	X	CFG_REG_SELECT: 0 = NONE 1 = HW 2 = SW
10	X	HW_RAMP_DONE: 0 = FALSE 1 = TRUE
9	X	SW_RAMP_DONE: 0 = FALSE 1 = TRUE
8	X	LONG_LATENCY_THROTTLE: 0 = DISABLE 1 = ENABLE
7	X	HW_RESTORE_EN: 0 = DISABLE 1 = ENABLE
6	X	HW_THROTTLE_EN: 0 = DISABLE 1 = ENABLE
5	X	SW_RAMP_STATUS: 0 = DONE 1 = INPROGRESS
4	X	HW_RAMP_STATUS: 0 = DONE 1 = INPROGRESS
3:2	X	SEQ_STATUS: 0 = SEQ_DIABLE 1 = SEQ_ENABLE 2 = SEQ_BUSY_FAST_MODE 3 = SEQ_BUSY_SLOW_MODE
1	X	SW_OVERRIDE_STATUS: 0 = DISABLE 1 = ENABLE
0	X	SWCTL_STATUS: 0 = DISABLE 1 = ENABLE

6.9.222 CLK_RST_CONTROLLER_SUPER_GR3D_CLK_DIVIDER_0

pulse skipper for gr3d_clk

Offset: 0x558 │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	DISABLE	SUPER_GR3DDIV_ENB: 0 = Disable divider. 0 = DISABLE 1 = ENABLE
30	DISABLE	SUPER_GR3DDIV_USE_THERM_CONTROLS: 1 = Use therm controls for pulse skipper 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_GR3DDIV_DIVIDEND: Actual value = n + 1.

Bit	Reset	Description
7:0	0x0	SUPER_GR3DDIV_DIVISOR: actual value = n + 1.

6.9.223 CLK_RST_CONTROLLER_SPARE_REG0_0

Offset: 0x55c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:2	0x0	VAL
1	0x0	TMR_CLKEN_STICKY
0	0x0	EMC_LATENCY_OVERRIDE

6.9.224 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_CORE_HOST_0

Offset: 0x600 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	XUSB_CORE_HOST_CLK_SRC: 000 = clk_m, 001 = pllP_out0, 010 = pllC2_out0, 011 = pllC_out0, 100 = pllC3_out0, 101 = pllRefe_OUT 0 = CLK_M 1 = PLLP_OUT0 2 = PLLC2_OUT0 3 = PLLC_OUT0 4 = PLLC3_OUT0 5 = PLLREFE_OUT
7:0	0x0	XUSB_CORE_HOST_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.225 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FALCON_0

Offset: 0x604 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	XUSB_FALCON_CLK_SRC: 000 = clk_m, 001 = pllP_out0, 010 = pllC2_out0, 011 = pllC_out0, 100 = pllC3_out0, 101 = pllRefe_OUT 0 = CLK_M 1 = PLLP_OUT0 2 = PLLC2_OUT0 3 = PLLC_OUT0 4 = PLLC3_OUT0 5 = PLLREFE_OUT
7:0	0x0	XUSB_FALCON_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.226 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_FS_0

Offset: 0x608 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	XUSB_FS_CLK_SRC: 00 = clk_m, 01 = FO_48M, 01 = pllP_out0, 10 = HSIC_480 0 = CLK_M 1 = FO_48M 2 = PLLP_OUT0 3 = HSIC_480
7:0	0x0	XUSB_FS_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.227 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_CORE_DEV_0

Offset: 0x60c | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	XUSB_CORE_DEV_CLK_SRC: 000 = clk_m, 001 = pllP_out0, 010 = pllC2_out0, 011 = pllC_out0, 100 = pllC3_out0, 101 = pllRefe_OUT 0 = CLK_M 1 = PLLP_OUT0 2 = PLLC2_OUT0 3 = PLLC_OUT0 4 = PLLC3_OUT0 5 = PLLREFE_OUT
7:0	0x0	XUSB_CORE_DEV_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.228 CLK_RST_CONTROLLER_CLK_SOURCE_XUSB_SS_0

Offset: 0x610 | Read/Write: R/W | Reset: 0x00000000 (0b000xxx00xxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	XUSB_SS_CLK_SRC: 000 = clk_m, 001 = pllRefe_OUT, 010 = clk_s, 011 = HSIC_480, 100 = pllC_out0, 101 = pllC2_out0, 110 = pllC3_out0, 111 = osc_div 0 = CLK_M 1 = PLLREFE_OUT 2 = CLK_S 3 = HSIC_480 4 = PLLC_OUT0 5 = PLLC2_OUT0 6 = PLLC3_OUT0 7 = OSC_DIV
25	0x0	XUSB_HS_CLK_BYPASS_SWITCH: 0 = HS clock is switch/dividers output 1 = HS Clock is derived from PLLU 60M output directly.
24	0x0	XUSB_SS_CLK_BYPASS_SWITCH: 0 = SS clock is switch/dividers output 1 = SS Clock is derived from osc_div clock directly.
7:0	0x0	XUSB_SS_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.229 CLK_RST_CONTROLLER_CLK_SOURCE_CILAB_0

Offset: 0x614 | Read/Write: R/W | Reset: 0xc0000000 (0b11xxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	CILAB_CLK_SRC: 00 = pllP_out0, 01 = pllC_out0, 10 = clk_m, 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 3 = CLK_M
7:0	0x0	CILAB_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.230 CLK_RST_CONTROLLER_CLK_SOURCE_CILCD_0

Offset: 0x618 | Read/Write: R/W | Reset: 0xc0000000 (0b11xxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	CILCD_CLK_SRC: 00 = pllP_out0, 01 = pllC_out0, 10 = clk_m, 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 3 = CLK_M
7:0	0x0	CILCD_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.231 CLK_RST_CONTROLLER_CLK_SOURCE_CILE_0

Offset: 0x61c | Read/Write: R/W | Reset: 0xc0000000 (0b11xxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	CILE_CLK_SRC: 00 = pllP_out0, 01 = pllC_out0, 10 = clk_m, 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 3 = CLK_M
7:0	0x0	CILE_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.232 CLK_RST_CONTROLLER_CLK_SOURCE_DSIA_LP_0

Offset: 0x620 | Read/Write: R/W | Reset: 0xc0000000 (0b11xxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	DSIA_LP_CLK_SRC: 00 = pllP_out0, 01 = pllC_out0, 10 = clk_m, 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 3 = CLK_M
7:0	0x0	DSIA_LP_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.233 CLK_RST_CONTROLLER_CLK_SOURCE_DSIB_LP_0

Offset: 0x624 | Read/Write: R/W | Reset: 0xc0000000 (0b11xxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:30	CLK_M	DSIB_LP_CLK_SRC: 00 = pllP_out0, 01 = pllC_out0, 10 = clk_m, 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 3 = CLK_M
7:0	0x0	DSIB_LP_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.234 CLK_RST_CONTROLLER_CLK_SOURCE_ENTROPY_0

Offset: 0x628 | Read/Write: R/W | Reset: 0x40000000 (0b01xxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
31:30	CLK_M	ENTROPY_CLK_SRC: 00 = pllP_out0, 01 = clk_m 0 = PLLP_OUT0 1 = CLK_M
8	0x0	ENTROPY_CLK_LOCK: 1 = Lock entire CLK_SOURCE_ENTROPY register, can only be unlocked by reset.
7:0	0x0	ENTROPY_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.235 CLK_RST_CONTROLLER_CLK_SOURCE_DVFS_REF_0

Offset: 0x62c | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DVFS_REF_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
28	0x1	DVFS_REF_MASTER_CLKEN: Reserved.
7:0	0x0	DVFS_REF_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.236 CLK_RST_CONTROLLER_CLK_SOURCE_DVFS_SOC_0

Offset: 0x630 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	DVFS_SOC_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
28	0x1	DVFS_SOC_MASTER_CLKEN: Reserved.
7:0	0x0	DVFS_SOC_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.237 CLK_RST_CONTROLLER_CLK_SOURCE_TRACECLKIN_0

Offset: 0x634 | Read/Write: R/W | Reset: 0xc0000000 (0b110xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	TRACECLKIN_CLK_SRC: 000 = pllP_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllM_out0, 110 = clk_m 0 = PLLP_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLM_OUT0 6 = CLK_M
7:0	0x0	TRACECLKIN_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.238 CLK_RST_CONTROLLER_CLK_SOURCE_ADX0_0

Offset: 0x638 | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	ADX0_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M
28	0x1	ADX0_MASTER_CLKEN: Reserved.
7:0	0x0	ADX0_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.239 CLK_RST_CONTROLLER_CLK_SOURCE_AMX0_0

Offset: 0x63c | Read/Write: R/W | Reset: 0xd0000000 (0b1101xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	AMX0_CLK_SRC: 000 = pllA_out0, 001 = pllC2_out0, 010 = pllC_out0, 011 = pllC3_out0, 100 = pllP_out0, 110 = clk_m 0 = PLLA_OUT0 1 = PLLC2_OUT0 2 = PLLC_OUT0 3 = PLLC3_OUT0 4 = PLLP_OUT0 6 = CLK_M
28	0x1	AMX0_MASTER_CLKEN: Reserved.
7:0	0x0	AMX0_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.240 CLK_RST_CONTROLLER_CLK_SOURCE_EMC_LATENCY_0

Offset: 0x640 | Read/Write: R/W | Reset: 0x60000000 (0b011xxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	CLK_M	EMC_LATENCY_CLK_SRC: 000 = pllM_out0, 001 = pllC_out0, 010 = pllP_out0, 011 = clk_m, 100 = pllM_out_for_emc un-divided PllM_out0, 101 = pllC2_out, 110 = pllC3_out, 111 = empty 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M 4 = PLLM_UD 5 = PLLC2_OUT0 6 = PLLC3_OUT0
7:0	0x0	EMC_LATENCY_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

6.9.241 CLK_RST_CONTROLLER_CLK_SOURCE_SOC_THERM_0

Offset: 0x644 | Read/Write: R/W | Reset: 0x40000000 (0b010xxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:29	pllP_out0	SOC_THERM_CLK_SRC: 000 = pllM_out0, 001 = pllC_out0, 010 = pllP_out0, 011 = pllA_out0, 100 = pllC2_out0, 101 = pllC3_out0 0 = pllM_out0 1 = pllC_out0 2 = pllP_out0 3 = pllA_out0 4 = pllC2_out0 5 = pllC3_out0
7:0	0x0	SOC_THERM_CLK_DIVISOR: N = Divide by (n+1). (lsb denotes 0.5x)

7.0 CL-DVFS

The CL_DVFS module contains control logic that provides a hardware mechanism for controlling the clock rate and power supply voltage of the fast CPU complex.

This section describes the closed-loop dynamic voltage and frequency scaling (CL-DVFS) registers. It is not intended to be a programming guide to CL-DVFS, as it is expected that NVIDIA supplied drivers will always be used with it. However, the register interface is documented to aid with understanding those drivers.

7.1 CL-DVFS Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

7.1.1 CL_DVFS_CTRL_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	DFLL_CTRL_MODE: DFFL Control Mode. 0 = DISABLE: Disabled. The ring oscillator does not run. 1 = ENABLE_OPEN_LOOP: In Enable Open Loop mode, the ring oscillator will run, but the control loop will remain inactive. 2 = ENABLE_CLOSED_LOOP: In Enable Closed Loop mode, the control loop is enabled and the I2C interface will continuously update the control output in order to maintain the desired frequency set in the DFLL_FREQ_REQ_xxxx fields.

7.1.2 CL_DVFS_CONFIG_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000006 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000110)

Bit	Reset	Description
7:0	0x6	DFLL_CONFIG_DIV_S: Refclk divider for setting DFLL control loop sample rate. Refclk is divided by 32x the value in this field

7.1.3 CL_DVFS_PARAMS_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x000000f0 (0bxxxxxxx000000000xxxxx00011110000)

Bit	Reset	Description
24	0x0	DFLL_PARAMS_CG_SCALE: If set to 1, reduces the overall loop gain by a factor of 8.
23:22	0x0	DFLL_PARAMS_FORCE_MODE: 0 = DISABLE : Disabled. The I2C control value is never forced during a frequency change. 1 = FIXED : In Fixed Delay mode, the I2C control value is forced for a fixed number of sample periods. 2 = AUTO: In Auto mode, the I2C control value is forced for a calculated number of sample periods.
21:16	0x0	DFLL_PARAMS_CF_PARAM: Length of time that a forced I2C control value will be applied after a frequency change if forcing was requested for that change. In Fixed Delay mode, it provides the number of sample periods to force the I2C control value. In Auto mode, the force time is equal to I2C control output delta * cf_param / 16
10:8	0x0	DFLL_PARAMS_CI_PARAM: Integral term gain in the control loop controls how a cycle deficit/surfeit after a frequency change is cleared. It temporarily raises or lowers the core voltage to allow the total clock cycle count to converge with the ideal number of cycles that should have been produced since the last frequency change request. 0 = DISABLE 1 = DIV2 2 = DIV4 : 3 = DIV8 4 = DIV16

Bit	Reset	Description
		5 = DIV32 : 6 = DIV64 7 = DIV128
7:0	0xf0	DFLL_PARAMS_CG_PARAM: Overall loop gain control (SIGNED value), controls the overall response time of the control loop

7.1.4 CL_DVFS_TUNE0_0

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

Bit	Reset	Description
23:16	0x0	DFLL_TUNE0_DLY_STK: Input bits to both coarse (4) and fine (4) tune the delay
15:8	0x0	DFLL_TUNE0_DLY_SRAM: Input bits to both coarse (3) and fine (5) tune the delay of the SRAM path.
7:0	0x0	DFLL_TUNE0_DLY_INV: Input bits to both coarse (3) and fine (5) tune the delay of the inverter path

7.1.5 CL_DVFS_TUNE1_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:11	0x0	DFLL_TUNE1_DLY_FINE: Input bits to tune the two phases of the clock. 8 bits to tune, and 1 bit to choose high vs. low
10:0	0x0	DFLL_TUNE1_DLY_WIRE: Input bits to both coarse (2) and fine (9) tune the delay of wire dominated path

7.1.6 CL_DVFS_FREQ_REQ_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x0000ff40 (0bxxx00000000000011111111101000000)

Bit	Reset	Description
28	0x0	DFLL_FREQ_REQ_FORCE_EN: Set to '1' to force I2C control output to initial value specified by the 'force_val' field
27:16	0x0	DFLL_FREQ_REQ_FORCE_VAL: Value forced onto the integrator during a frequency transition, only used if the 'force_en' field below is '1' AND the global 'force_mode' field of the dfll_params register is not set to 'disable'. The best value is (desired I2C control value - safe I2C control value) x 128) / Cg.
15:8	0xff	DFLL_FREQ_REQ_SCALE: Proportion of output clock cycles (+1) to *not* skip over a period of 256 cycles
7	0x0	DFLL_FREQ_REQ_VALID: Set to '1' to indicate that the frequency configuration is valid and should be used. If this bit is '0', the control loop will revert to 'open loop' mode, and the I2C control interface value will be determined by the 'safe' value.
6:0	0x40	DFLL_FREQ_REQ_MULT: Primary frequency multiplication factor 'F'. The ring oscillator output frequency will be (REF_CLK/2) x F.

7.1.7 CL_DVFS_SCALE_RAMP_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	DFLL_OUTPUT_RAMP_RATE: The ramp up/ramp down rate of the control signal to the output scaler. Determines the number of cycles (+1) to wait for each counter step.

7.1.8 CL_DVFS_DROOP_CTRL_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000810 (0bxxxxxxxx00000000xxxx100000010000)

Bit	Reset	Description
23:16	0x0	DFLL_DROOP_CTRL_MIN_FREQ: The minimum allowed ring oscillator frequency before the 'droop' clock skipper is enabled. The value written determines the minimum number of cycles that are allowed in 4 REF_CLK cycles' Fmin = droop_ctrl.min_freq * (REF_CLK / 4)
11:8	0x8	DFLL_DROOP_CTRL_CUT: CPU clock is scaled by (cut+1)/16 immediately after reaching the minimum ring oscillator frequency
7:0	0x10	DFLL_DROOP_CTRL_RATE: Controls the rate at which clock cycles are re-introduced to the droop skipper after it has been ramped down to compensate for a frequency droop. It is the number of cycles (+1) to wait for each counter step

7.1.9 CL_DVFS_OUTPUT_CFG_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x50200000 (0bx1010000xx100000xx00000000000000)

Bit	Reset	Description
30	0x1	DFLL_OUTPUT_CONFIG_I2C_ENABLE: Master enable control for I2C control value updates. If this field is '0', then I2C control messages are inhibited, regardless of the DFLL mode.
29:24	0x10	DFLL_OUTPUT_CONFIG_SAFE: 'Safe' value for the OUTPUT control interface. This value will be output whenever OUTPUT is enabled but the DFLL is disabled, or in open loop mode.
21:16	0x20	DFLL_OUTPUT_CONFIG_MAX: Maximum allowed value on the OUTPUT control interface.
13:8	0x0	DFLL_OUTPUT_CONFIG_MIN: Minimum allowed value on the OUTPUT control interface.
7	0x0	DFLL_OUTPUT_CONFIG_DELTA_EN: 1: In conjunction with 'clk_en'=1, causes the PWM clock/data output to only become active (for 32 cycles) whenever a change in the PWM value occurs. 0: The PWM data/clock outputs run continuously when enabled.
6	0x0	DFLL_OUTPUT_CONFIG_CLK_EN: Enables the PMIC control clock output for digitally controlled PMICs.
5:0	0x0	DFLL_OUTPUT_CONFIG_DIV_D: Divider setting for PWM PMIC control output (divides the SOC clock).

7.1.10 CL_DVFS_OUTPUT_FORCE_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx)

Bit	Reset	Description
6	0x0	DFLL_OUTPUT_FORCE_ENABLE: Enable the force value onto the OUTPUT control output
5:0	X	DFLL_OUTPUT_FORCE_VALUE: Value to force OUTPUT control output whenever the i2c_ctrl_force_enable field is set to 1.

7.1.11 CL_DVFS_MONITOR_CTRL_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2:0	X	DFLL_MONITOR_CTRL_SELECT: Selects a control loop data source for monitoring 0 = DISABLE 1 = CYCLE_INT 2 = PRO_TERM 3 = INT_TERM 4 = OUTPUT_INT 5 = OUTPUT_VALUE 6 = FREQ

7.1.12 CL_DVFS_MONITOR_DATA_0

Offset: 0x2c | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
16	X	DFLL_MONITOR_DATA_NEW: Set to '1' whenever a new sample is generated. Cleared on reads.
15:0	X	DFLL_MONITOR_DATA_VAL: Read data monitor source selected by dfll_monitor_ctrl_select. If monitoring is enabled, this field is updated every sample period.

7.1.13 CL_DVFS_I2C_CFG_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x0010fXXX (0bxxxxxxxxxxxx1x0001111xxxxxxxxxxx)

Bit	Reset	Description
20	0x1	I2C_BUS_ARB: Enables arbitration for bus
18:16	0x0	I2C_MASTER_CODE: Master code for high-speed transfers
15	0x1	I2C_PACKET_MODE: Enables Packet mode for high-speed transfers
14:12	0x7	I2C_SIZE: Size of voltage values from 1 to 7
10	X	I2C_ADDR_7BIT_10BIT: 0: Selects 7-bit addressing 1: Select 10-bit addressing
9:0	X	I2C_SLAVE_ID: External slave ID address

7.1.14 CL_DVFS_I2C_VDD_REG_ADDR_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	I2C_DEFAULT_DATA: Default data
7:0	X	I2C_ADDR_DATA: Address for voltage sel

7.1.15 CL_DVFS_I2C_STS_0

Offset: 0x48 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6:1	X	I2C_LAST_VALUE: Output value from the DFLL of last I2C request that was completed successfully
0	X	I2C_REQ_PENDING: Indicates there is an outstanding I2C request

7.1.16 CL_DVFS_INTR_STS_0

Offset: 0x5c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	MAX_VDD_LIMIT_INTR: Interrupt to indicate the DFLL has hit the VDD ceiling as programmed in DFLL_OUTPUT_CONFIG_MAX
0	X	MIN_VDD_LIMIT_INTR: Interrupt to indicate the DFLL has hit the VDD floor as programmed in DFLL_OUTPUT_CONFIG_MIN

7.1.17 CL_DVFS_INTR_EN_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	MAX_VDD_LIMIT_INTR_EN: Enable for interrupt to indicate the DFLL has hit the VDD ceiling as programmed in DFLL_OUTPUT_CONFIG_MAX
0	X	MIN_VDD_LIMIT_INTR_EN: Enable for interrupt to indicate the DFLL has hit the VDD floor as programmed in DFLL_OUTPUT_CONFIG_MIN

7.1.18 CL_DVFS_I2C_CLK_DIVISOR_REGISTER_0

The divisor values (N) must be programmed so that:

- SCL frequency (Std/Fast mode) = ClkSourceFreq/(8*(N+1))
- SCL frequency (HS mode) = ClkSourceFreq/(12*(N+1))

Offset: 0x16c | Read/Write: R/W | Reset: 0x00190001 (0b00000000000110010000000000000001)

Bit	Reset	Description
31:16	0x19	I2C_CLK_DIVISOR_STD_FAST_MODE:N = Divide by n+1
15:0	0x1	I2C_CLK_DIVISOR_HSMODE:N = Divide by n+1

7.1.19 CL_DVFS_OUTPUT_LUT_0

This lookup table (LUT) contains voltage LUT values stored in a 33-deep by 8 bit wide RAM. To program the LUT, simply sequentially address the RAM, using the base offset address of 10’b10xxxxxxx. Addresses are DWORD aligned, not BYTE aligned. PMIC values must be programmed to span the entire desired operating range, and be monotonically increasing. The logic assumes that the middle entry (16th) corresponds to the ‘safe’ initial voltage, as specified in the dfll_output_config.safe field.

[THIS PAGE INTENTIONALLY LEFT BLANK]

8.0 TIMERS

This section documents the various timers available to software in a Tegra® 4 series system. The following table summarizes these timers.

Table 22: List of Timers

Name	Primary Use	Related Interrupt	Secure	Freq.	Notes
RTC	Wall Clock Timer	RTC	Pseudo	32kHz	See the Real-Time Clock section in this TRM for detailed information.
TMR	NVIDIA® Generic Timers	TMR9-0	Cfg	1MHz	Configurable to be secure
WDT	TMR timers used as Watchdog	WDT_<>	Cfg	1MHz	Configurable to be secure
TSC	Reference for GIT	N/A	Yes	OSC	Counter value can only be updated in secure mode.
GIT	ARM CPU Generic Timers	PPIs*	Yes	TSC	These timers use TSC as reference.

*PPIs are per CPU Private Peripheral Interrupts

8.1 ARM CPU Generic Timers (GITs)

The ARM® Generic Timer architecture defines generic CPU timer requirements. The architecture requires that the SoC supply a timestamp counter reference that is independent of the CPU clock frequency.

The Cortex™-A15 Generic Timer includes:

- A physical counter that contains the count value of the system counter.
- A virtual counter that indicates virtual time. The virtual counter contains the value of the physical counter minus a 64-bit virtual offset.
- A set of four timers per CPU.

The four timers are the following:

- Secure Physical Timer
- Non-Secure Physical Timer
- Hypervisor Timer
- Virtual Timer

Refer to the ARM Architecture Reference Manual v7 for generic timer specifications.

8.2 Generic Timer System Counter (TSC)

The ARM Generic Timer Specification requires a system time-stamp counter (TSC) supplied by the SoC.

The TSC is implemented as part of the Tegra 4 Power Management Controller (PMC). It is a 56-bit counter which runs at the crystal oscillator clock frequency. Refer to the Power Management Controller section in this TRM for additional TSC information.

8.3 NVIDIA Timers (TMR)

The programmable timer block contains ten 29-bit programmable timer counters and one 32-bit timestamp counter.

Timer interval, one-shot, or periodic interrupts are configured in the Timer Present Value register (PTV). When enabled, the timer loads the Timer Present Value count and begins decrementing every one microsecond. The timer generates a timer request when the count reaches zero.

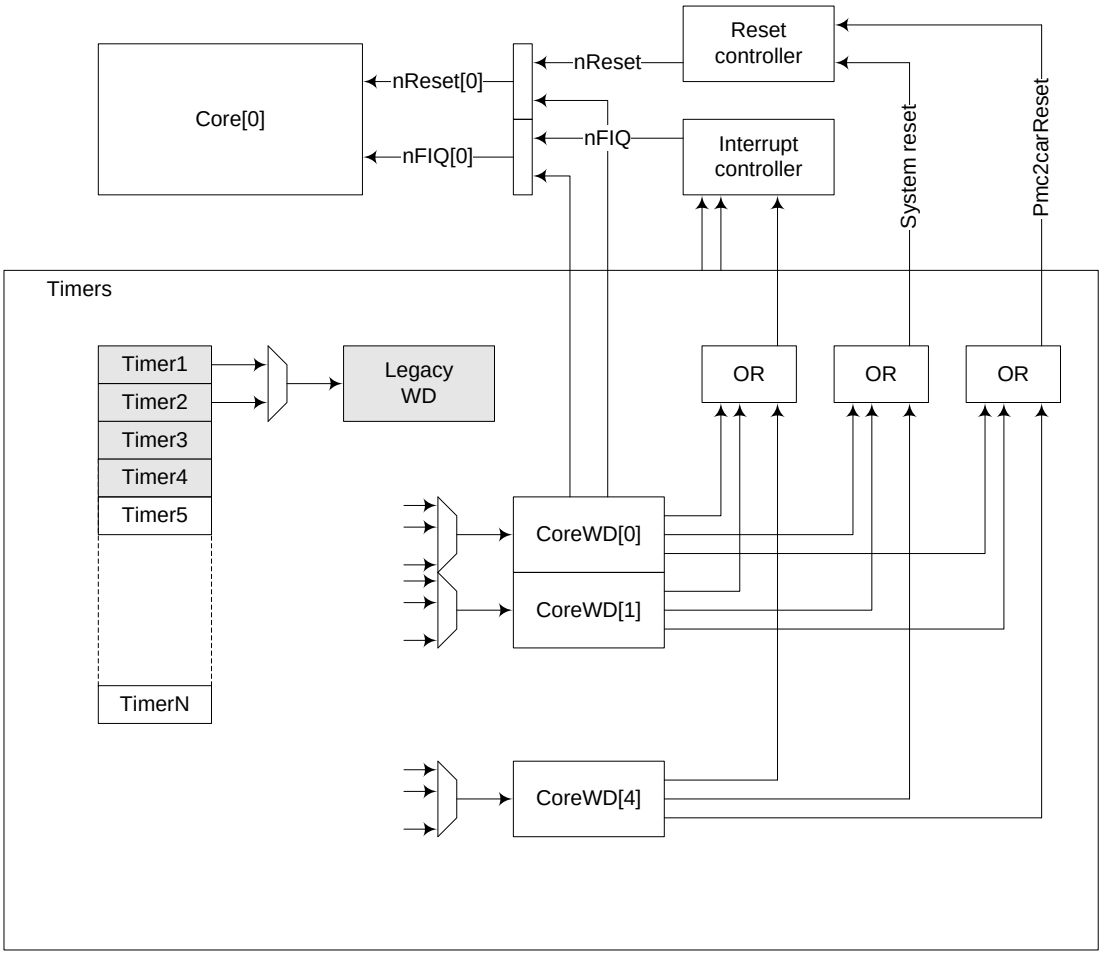
If the PTV periodic control bit is set, the timer reloads the PTV interval and continues decrementing the timer.

The timer count is 29 bits wide, supporting timer intervals of 16.384 ms maximum or 1 µs minimum. Bit 30 of the periodic Timer Value (PTV) is the enable bit for the timer. Reading the PTV Register clears any pending timer interrupt.

A read-only Preset count register (PCR) allows the processor to inspect the current value of the timer decrement counter. The timestamp counter (32 bits wide) is cleared to zero on reset, and counts upwards every 1 µs.

The figure below shows a reference decomposition of the timers in functional blocks.

Figure 19: Timer Functional Block Diagram



- The timers are numbered 0 to 9
- Timers 1 to 5 have a dedicated interrupt bit
- Timers 0, 6, 7, 8, and 9 share a common interrupt called SharedTimerInterrupt.
- Timer interrupts are assigned as follows:
  - Timer 1 = primary interrupt controller bit 0

- Timer 2 = primary interrupt controller bit 1
- Timer 3 = secondary interrupt controller bit 9
- Timer 4 = secondary interrupt controller bit 10

8.4 Watchdog Timers (WDTs)

A watchdog timer facilitates recovery from system lockup conditions. The watchdog timer can interrupt the processor when the selected timer counter expires. Under normal operation, this interrupt is serviced by reading the timer status register (TIMER_WDTx_STATUS). Any timer in the Tegra 4 timer block may be configured as a watchdog timer.

- The Tegra 4 series processor provides five WatchDog Timers, one each for main CPUs and one for COP (AVP): WDT0 is allocated to CPU0 of cluster0 or the shadow CPU of cluster1
- WDT1 is allocated to CPU1
- WDT2 is allocated to CPU2
- WDT3 is allocated to CPU3
- WDT4 is allocated to COP (AVP)

The 5 watchdog timers are directly associated with a processor core, i.e., 4 CPU cores[1] and the COP:

- The watchdog is reset by the corresponding core reset signal
- WD timers 0 to 3 are associated with CPU 0 to 3, WD 4 is associated with COP

There are no dedicated timers for the WDTs. Each WDT can be configured to use any one of the TMRs as its timer source.

Based on the configuration of the TIMER_WDTx_CONFIG register, a WDT works as follows:

- When the timer decrements to zero the first time, a WDT interrupt can be generated
- When the timer consecutively decrements down to zero a third time, without the processor reading the interrupt status register (TIMER_TMRx_TMR_PCR), a per CPU reset can be generated.
- When the timer consecutively decrements down to zero a fourth time, a system reset can be generated. The cause of this system reset can be read by reading PMC_RST_STATUS[RST_SOURCE] (see the Power Management Controller section for more information).

8.5 Secure TMRs and Secure WDTs

The Tegra 4 series processor TMRs and/or WDTs can be configured to be secure timers. As described in TIMER_SECURE_CFG register table later in this section, a secure register TIMER_SECURE_CFG -- which itself can only be written by secure writes – can be configured such that each of the TMRs and WDTs can be independently configured to be secure. By default all of TMRs and WDTs are non-secure and backward compatible.

8.6 Watchdog Timer Programming Guide

The watchdog timer can be programmed to reset just the CPU, just the AVP, or the entire system.

If only the CPU is reset, then AVP will just continue running as-is. The CPU reset vector should be programmed in advance so that it skips the iROM boot code and jumps directly into its own CPU boot routine (which can be anywhere.)

If only the AVP is reset, the same is true. The CPU keeps running as-is. If the AVP reset vector is set to default, it runs the normal boot code. It is up to the boot loader to decide if it needs to restart the CPU as well.

[1] CPU Core 0 is treated in a specific manner so that the watchdog process is transparent to software migration from the G to the LP process.

The watchdog timer can be programmed to the maximum timeout value of 0x3fffffff. At 1 µs/count, this is more than one thousand seconds.

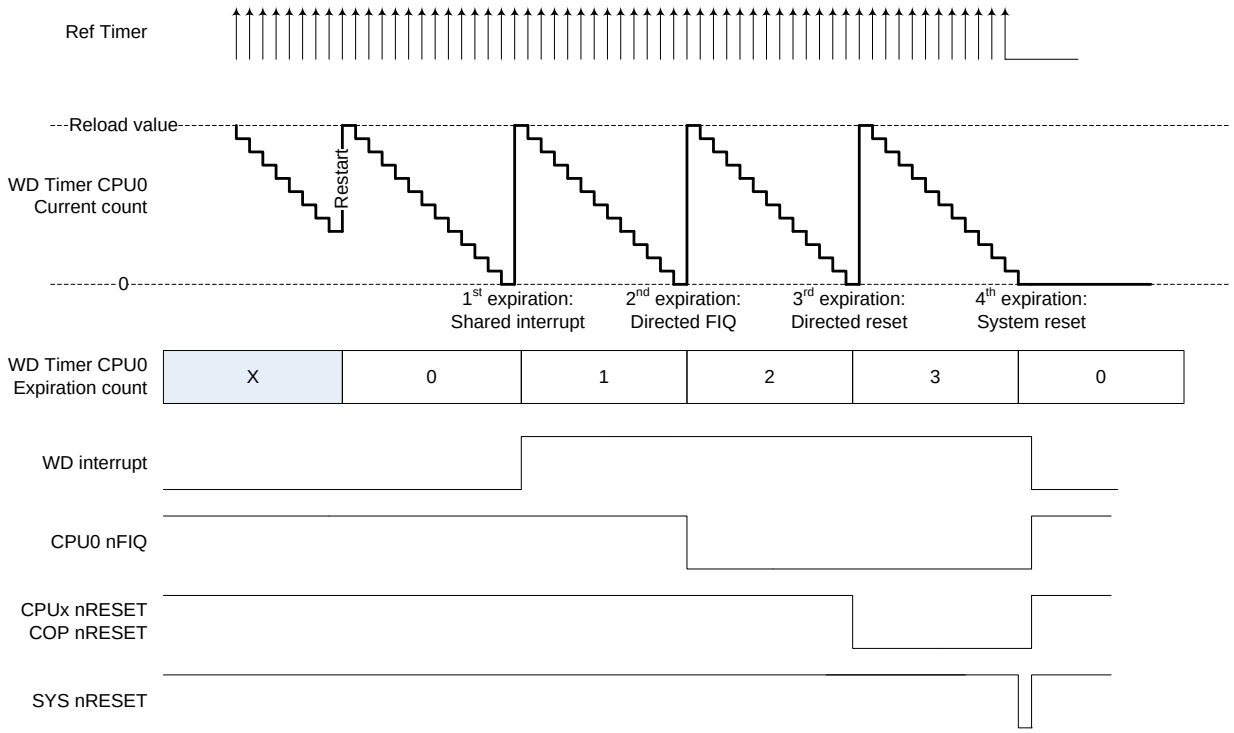
It is operated as follows:

1. Watchdog generates an Interrupt based on the Timer
    a. TMR1 or TMR2 can be selected as source
    b. Timer should be programmed as Periodic with Interrupt.
    c. CAR.RST_SOURCE.WDT_EN = 1
    d. CAR.RST_SOURCE.WDT_SEL selects TMR2/TMR1
2. Reset can be for the AVP, CPU, or full system
    a. CAR.RST_SOURCE.WDT_***_RST_EN = 1
3. Upon reboot, CAR.RST_SOURCE.WDT_*_RST_STA indicates the cause of the reset

The five watchdog timers operate as follows:

- Select any one of the general-purpose timers as timing reference
- Standard down counter with programmable period, that also counts expirations, specific action are taken at the counter expiration
    - If the expiration count is 0, a normal interrupt may be generated (probably pooled)
    - If the expiration count is 1, the FIQ for the corresponding processor may be asserted, this operation bypasses the normal interrupt controller
    - If the expiration count is 2, the reset for a programmable set of processors is asserted, this may be the null set
    - If the expiration count is 3, a system wide reset may be asserted (two variants, standard system or closer to external hardware reset)

Figure 20: Watchdog Example Timing Diagram

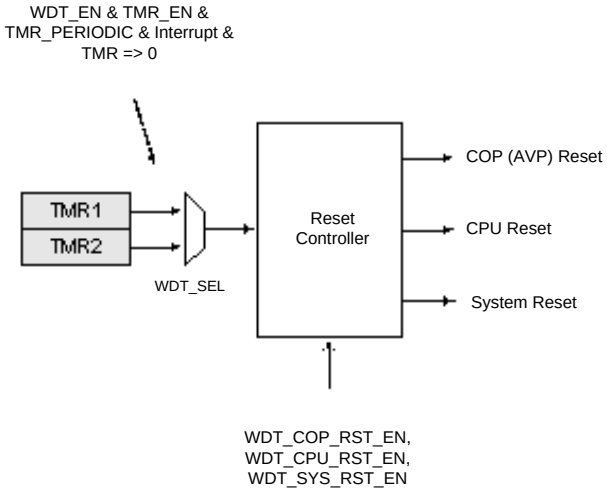


Any operation that targets CPU0 is done in the following way:

- Any output is broadcast to both instances of CPU0, in CPU cluster 0 (G) and 1 (LP). This applies to the reset and FIQ signals.
- Resetting any of the CPU0 results in resetting the WD associated with CPU 0

Although the timing diagram implies that you need to restart the timer when you service the interrupt, this is not necessary as long as you can always service the interrupt (clear the TMR interrupt) before the next timeout occurs. The reset only happens if a new timeout occurs when the TMR interrupt is still active.

Figure 21: Watchdog Block Diagram



8.7 Timers Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

8.7.1 TIMER_TMR1_TMR_PTV_0

Before programming the timer:

- Input oscillator frequency must be specified in the TIMERUS_USEC_CFG register.
- Program the number of 1 µs timer counts per "tick" in the PTV (Present Trigger Value) register.
- Set the PTV EN (enable bit) to start counting down.
- When the count reaches zero, an interrupt is generated.
- To auto-reload the timer counter, set the PTV PER (periodic) bit. Otherwise, leave it clear for a one-shot.

Timer Present Trigger Value (Set) Register

Offset: 0x0 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000000000000000000)

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE
28:0	0x0	TMR_PTV: Trigger Value: count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

8.7.2 TIMER_TMR1_TMR_PCR_0

When interrupt is generated, write "1" to PCR[30] to clear the interrupt

Timer Present Count Value (Status) Register

Offset: 0x4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no affect. Write-1-to-Clear
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

8.7.3 TIMER_TMR2_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000000000000000000)

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE
28:0	0x0	TMR_PTV: Trigger Value: count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

8.7.4 TIMER_TMR2_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 0xc | Read/Write: R/W | Reset: 0xXXXXXXXX (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no affect. Write-1-to-Clear
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

8.7.5 TIMER_TMR3_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000000000000000000)

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28:0	0x0	TMR_PTV: Trigger Value: count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

8.7.6 TIMER_TMR3_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 0x54 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no effect. Write-1-to-Clear
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

8.7.7 TIMER_TMR4_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000000000000000000)

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE
28:0	0x0	TMR_PTV: Trigger Value: count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

8.7.8 TIMER_TMR4_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 0x5c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no effect. Write-1-to-Clear
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

8.7.9 TIMER_TMR5_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE
28:0	0x0	TMR_PTV: Trigger Value: count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

8.7.10 TIMER_TMR5_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 0x64 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no effect. Write-1-to-Clear
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

8.7.11 TIMER_TMR6_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 0x68 | Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE
28:0	0x0	TMR_PTV: Trigger Value: Count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

8.7.12 TIMER_TMR6_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 0x6c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no effect. Write-1-to-Clear

Bit	R/W	Reset	Description
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

8.7.13 TIMER_TMR7_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 0x70 | Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE
28:0	0x0	TMR_PTV: Trigger Value: count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

8.7.14 TIMER_TMR7_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 0x74 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no effect. Write-1-to-Clear
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

8.7.15 TIMER_TMR8_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 0x78 | Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE
28:0	0x0	TMR_PTV: Trigger Value: count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

8.7.16 TIMER_TMR8_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 0x7c| Read/Write: R/W | Reset: 0xXXXXXXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no effect. Write-1-to-Clear
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

8.7.17 TIMER_TMR9_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000000000000000000)

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE
28:0	0x0	TMR_PTV: Trigger Value: count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

8.7.18 TIMER_TMR9_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 0x84 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no effect. Write-1-to-Clear
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

8.7.19 TIMER_TMR0_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000000000000000000)

Bit	Reset	Description
31	0x0	EN: Enable Timer 0 = DISABLE 1 = ENABLE
30	0x0	PER: Enable Periodic Interrupt 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28:0	0x0	TMR_PTV: Trigger Value: Count trigger value (count length). This is in n+1 scheme. If you program the value n, the count trigger value will actually be n+1.

8.7.20 TIMER_TMR0_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 0x8c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no effect. Write-1-to-Clear
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

8.8 Timer USEC CFG

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

The purpose of the USEC_CFG/CNTR_1US registers is to provide a fixed time base (in microseconds) to be used by the rest of the system regardless of the oscillator frequency (i.e., 12 MHz, 13 MHz, 19.2 MHz, 26 MHz, or other frequencies).

8.8.1 USEC_CFG Register

Software should first configure this register by telling what fraction of 1 microsecond each oscillator clock represents. For example, if the oscillator clock is running at 12 MHz, then each oscillator clock represents 1/12 of a microsecond. "USEC_DIVIDEND" and "USEC_DIVISOR" are used to indicate what fraction of 1 microsecond each oscillator clock represents.

Table 23: Oscillator Clock Frequency Indicator for USEC

Oscillator Clock Frequency	Dividend/Divisor	USEC_DIVIDEND/USEC_DIVISOR
12 MHz	1/12	0x00 / 0x0b
13 MHz	1/13	0x00 / 0x0c
19.2 MHz	5/96	0x04 / 0x5f
26 MHz	1/26	0x00 / 0x19
16.8 MHz	5/84	0x04 / 0x53
38.4 MHz	5/192	0x04 / 0xbf
48 MHz	1/48	0x00 / 0x2f

8.9 CNTR_1US Register

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

This free-running read-only register/counter changes once every microsecond and is used mainly by hardware (can also be used by software). It starts counting from 0 once it is out of system reset and will continue counting forever, unless the oscillator clock is stopped or during a system reset.

(A) Software Use

Although there's no interrupt mechanism for this register, software can read the content of this register multiple times to determine the amount of time that has elapsed.

(B) Hardware Use

- Used by the 4 timers to determine whether the programmed timer value has been reached; these 4 timers can trigger interrupts.
- To provide a periodic USEC pulse to be used by the flow controller to count the programmable number of microseconds before a flow control condition is triggered.
- Used by secure boot logic.

8.9.1 TIMERUS_CNTR_1US_0

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

Bit	Reset	Description
31:16	x	HIGH_VALUE: Elapsed time in microseconds
15:0	x	LOW_VALUE: Elapsed time in microseconds

8.9.2 TIMERUS_USEC_CFG_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x0000000c (0bxxxxxxxxxxxxxxxx0000000000001100)

Bit	Reset	Description
15:8	0x0	USEC_DIVIDEND: microsecond dividend. (n+1)
7:0	0xc	USEC_DIVISOR: microsecond divisor. (n+1)

8.9.3 TIMERUS_CNTR_FREEZE_0

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	DBG_FREEZE_COP: 1 = freeze timers when COP is in debug state, 0 = no freeze.
3	0x0	DBG_FREEZE_CPU3: 1 = freeze timers when CPU3 is in debug state, 0 = no freeze.
2	0x0	DBG_FREEZE_CPU2: 1 = freeze timers when CPU2 is in debug state, 0 = no freeze.
1	0x0	DBG_FREEZE_CPU1: 1 = freeze timers when CPU1 is in debug state, 0 = no freeze.
0	0x0	DBG_FREEZE_CPU0: 1 = freeze timers when CPU0 is in debug state, 0 = no freeze.

8.10 Watchdog Timers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

The following register offsets are relative to the TMR block base.

Each Watchdog timer has the following:

- CoreWatchdog Configuration Register
- Timer sources are numbered 1 through 10 (0 is not used and results in undefined behavior)
- Period of 0 results in MAX period
- The generated interrupt is dependent on the WD index.
- WD0 to 3 generate a CPU_WD_Interrupt,
- WD4 generates a COP_WD_Interrupt
- CPU0 settings apply to both clusters
- Core numbering: 0-3 applies to CPU0-CPU3, 4 applies to COP

8.10.1 TIMER_WDT0_CONFIG_0

Core Watchdog Configuration Register

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20:16	0x0	CoreResetBitmapEn: Enable reset of a set of cores at third expiration counter (one bit per proc) same level the signal from PMC asserted in case of external reset.
15	0x0	Pmc2CarResetEn: Enable Full system reset at fourth expiration of the counter 0 = DISABLE 1 = ENABLE
14	0x0	SystemResetEnable: Enable system reset at fourth expiration of the counter, at same level as existing watchdog interrupt 0 = DISABLE 1 = ENABLE
13	0x0	FIQEnable: Enable FIQ assertion at second expiration of the counter 0 = DISABLE 1 = ENABLE
12	0x0	InterruptEnable: Enable interrupt at first expiration of the counter 0 = DISABLE 1 = ENABLE
11:4	0x0	Period: Measured in periods of selected timer, this is reload value
3:0	0x0	TimerSource: Select the timer used as reference (1..10) 0 = TMR0 1 = TMR1 2 = TMR2 3 = TMR3 4 = TMR4 5 = TMR5 6 = TMR6 7 = TMR7 8 = TMR8 9 = TMR9

8.10.2 TIMER_WDT0_STATUS_0

Core Watchdog Status

Offset: 0x104 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13:12	X	CurrentExpirationCount: Current count of expiration since the last start of expiration
11:4	X	CurrentCount: Current value of the counter
2	X	FiqStatus: Current status of FIQ, cleared by start
1	X	InterruptStatus: Current status of interrupt, cleared by start
0	X	Enabled: 1 when counter is active. When true, writes to corresponding config reg are ignored

8.10.3 TIMER_WDT0_COMMAND_0

The StartCounter bit enables watchdog counter operation, loads the watchdog counter, starts the watchdog timer to count down, resets the expiration count to 0, and clears all flags. Also used as restart.

The counter can be disabled by setting the DisableCounter bit, but only if the unlock register has been programmed before with the correct pattern. Writing to the command register always clears the disable unlock register. When set while StartCounter is 0 and the unlock register contains the unlock pattern the Watchdog transitions back to disabled.

The register fields are Write to set only.

CoreWatchdogCommand Register

Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	DisableCounter: Write 1 to disable WDT counter if unlocked
0	0x0	StartCounter: Write 1 to enable WDT counter

8.10.4 TIMER_WDT0_UNLOCK_PATTERN_0

CoreWatchdogDisableUnlock Register

Offset: 0x10c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	UnlockPattern: Must be written with value 0xC45A to allow write to DisableCounter to take effect. The pattern is reset at each write to Command Reg.

8.10.5 TIMER_WDT1_CONFIG_0

Offset: 0x120 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20:16	0x0	CoreResetBitmapEn: Enable reset of a set of cores at the third expiration counter (one bit per proc) same level the signal from PMC asserted in case of external reset.

Bit	Reset	Description
15	0x0	Pmc2CarResetEn: Enable Full system reset at the fourth expiration of the counter, at 0 = DISABLE 1 = ENABLE
14	0x0	SystemResetEnable: Enable system reset at the fourth expiration of the counter, at same level as existing watchdog interrupt 0 = DISABLE 1 = ENABLE
13	0x0	FIQEnable: Enable FIQ assertion at the second expiration of the counter 0 = DISABLE 1 = ENABLE
12	0x0	InterruptEnable: Enable interrupt at the first expiration of the counter 0 = DISABLE 1 = ENABLE
11:4	0x0	Period: Measured in periods of selected timer, this is the reload value
3:0	0x0	TimerSource: Select the timer used as reference (1..10) 0 = TMR0 1 = TMR1 2 = TMR2 3 = TMR3 4 = TMR4 5 = TMR5 6 = TMR6 7 = TMR7 8 = TMR8 9 = TMR9

8.10.6 TIMER_WDT1_STATUS_0

CoreWatchdogStatus

Offset: 0x124 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13:12	X	CurrentExpirationCount: Current count of expiration since the last start of expiration
11:4	X	CurrentCount: Current value of the counter
2	X	FiqStatus: Current status of FIQ, cleared by start
1	X	InterruptStatus: Current status of interrupt, cleared by start
0	X	Enabled: 1 when counter is active. When true, writes to corresponding config reg are ignored

8.10.7 TIMER_WDT1_COMMAND_0

CoreWatchdogCommand Register

Offset: 0x128 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	DisableCounter: Write 1 to disable WDT counter if unlocked
0	0x0	StartCounter: Write 1 to enable WDT counter

8.10.8 TIMER_WDT1_UNLOCK_PATTERN_0

CoreWatchdogDisableUnlock Register

Offset: 0x12c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	UnlockPattern: Must be written with value 0xC45A to allow write to DisableCounter to take effect. The pattern is reset at each write to Command Reg.

8.10.9 TIMER_WDT2_CONFIG_0

Offset: 0x140 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20:16	0x0	CoreResetBitmapEn: Enable reset of a set of cores at the third expiration counter (one bit per proc) same level the signal from PMC asserted in case of external reset.
15	0x0	Pmc2CarResetEn: Enable Full system reset at the fourth expiration of the counter, at 0 = DISABLE 1 = ENABLE
14	0x0	SystemResetEnable: Enable system reset at the fourth expiration of the counter, at same level as existing watchdog interrupt 0 = DISABLE 1 = ENABLE
13	0x0	FIQEnable: Enable FIQ assertion at the second expiration of the counter 0 = DISABLE 1 = ENABLE
12	0x0	InterruptEnable: Enable interrupt at the first expiration of the counter 0 = DISABLE 1 = ENABLE
11:4	0x0	Period: Measured in periods of selected timer, this is reload value
3:0	0x0	TimerSource: Select the timer used as reference (1..10) 0 = TMR0 1 = TMR1 2 = TMR2 3 = TMR3 4 = TMR4 5 = TMR5 6 = TMR6 7 = TMR7 8 = TMR8 9 = TMR9

8.10.10 TIMER_WDT2_STATUS_0

CoreWatchdogStatus

Offset: 0x144 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13:12	X	CurrentExpirationCount: Current count of expiration since the last start of expiration
11:4	X	CurrentCount: Current value of the counter

Bit	Reset	Description
2	X	FiqStatus: Current status of FIQ, cleared by start
1	X	InterruptStatus: Current status of interrupt, cleared by start
0	X	Enabled: 1 when counter is active. When true, writes to corresponding config reg are ignored

8.10.11 TIMER_WDT2_COMMAND_0

CoreWatchdogCommand Register

Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	DisableCounter: Write 1 to disable WDT counter if unlocked
0	0x0	StartCounter: Write 1 to enable WDT counter

8.10.12 TIMER_WDT2_UNLOCK_PATTERN_0

CoreWatchdogDisableUnlock Register

Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	UnlockPattern: Must be written with value 0xC45A to allow write to DisableCounter to take effect. The pattern is reset at each write to Command Reg.

8.10.13 TIMER_WDT3_CONFIG_0

CoreWatchdogConfiguration Register

Offset: 0x160 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20:16	0x0	CoreResetBitmapEn: Enable reset of a set of cores at the third expiration counter (one bit per proc) same level the signal from PMC asserted in case of external reset.
15	0x0	Pmc2CarResetEn: Enable Full system reset at the fourth expiration of the counter, at 0 = DISABLE 1 = ENABLE
14	0x0	SystemResetEnable: Enable system reset at the fourth expiration of the counter, at same level as existing watchdog interrupt 0 = DISABLE 1 = ENABLE
13	0x0	FIQEnable: Enable FIQ assertion at the second expiration of the counter 0 = DISABLE 1 = ENABLE
12	0x0	InterruptEnable: Enable interrupt at first expiration of the counter 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11:4	0x0	Period: Measured in periods of selected timer, this is reload value
3:0	0x0	TimerSource: Select the timer used as reference (1..10) 0 = TMR0 1 = TMR1 2 = TMR2 3 = TMR3 4 = TMR4 5 = TMR5 6 = TMR6 7 = TMR7 8 = TMR8 9 = TMR9

8.10.14 TIMER_WDT3_STATUS_0

CoreWatchdogStatus

Offset: 0x164 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13:12	X	CurrentExpirationCount: Current count of expiration since the last start of expiration
11:4	X	CurrentCount: Current value of the counter
2	X	FiqStatus: Current status of FIQ, cleared by start
1	X	InterruptStatus: Current status of interrupt, cleared by start
0	X	Enabled: 1 when counter is active. When true, writes to corresponding config reg are ignored

8.10.15 TIMER_WDT3_COMMAND_0

CoreWatchdogCommand Register

Offset: 0x168 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	DisableCounter: Write 1 to disable WDT counter if unlocked
0	0x0	StartCounter: Write 1 to enable WDT counter

8.10.16 TIMER_WDT3_UNLOCK_PATTERN_0

CoreWatchdogDisableUnlock Register

Offset: 0x16c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	UnlockPattern: Must be written with value 0xC45A to allow write to DisableCounter to take effect. The pattern is reset at each write to the Command Register.

8.10.17 TIMER_WDT4_CONFIG_0

Offset: 0x180 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20:16	0x0	CoreResetBitmapEn: Enable reset of a set of cores at third expiration counter (one bit per proc), same level the signal from PMC asserted in case of external reset.
15	0x0	Pmc2CarResetEn: Enable Full system reset at fourth expiration of the counter, at 0 = DISABLE 1 = ENABLE
14	0x0	SystemResetEnable: Enable system reset at fourth expiration of the counter, at same level as existing watchdog interrupt 0 = DISABLE 1 = ENABLE
13	0x0	FIQEnable: Enable FIQ assertion at second expiration of the counter 0 = DISABLE 1 = ENABLE
12	0x0	InterruptEnable: Enable interrupt at first expiration of the counter 0 = DISABLE 1 = ENABLE
11:4	0x0	Period: Measured in periods of selected timer, this is reload value
3:0	0x0	TimerSource: Select the timer used as reference (1..10) 0 = TMR0 1 = TMR1 2 = TMR2 3 = TMR3 4 = TMR4 5 = TMR5 6 = TMR6 7 = TMR7 8 = TMR8 9 = TMR9

8.10.18 TIMER_WDT4_STATUS_0

CoreWatchdogStatus

Offset: 10x84 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13:12	X	CurrentExpirationCount: Current count of expiration since the last start of expiration
11:4	X	CurrentCount: Current value of the counter
2	X	FiqStatus: Current status of FIQ, cleared by start
1	X	InterruptStatus: Current status of interrupt, cleared by start
0	X	Enabled: 1 when counter is active. When true, writes to corresponding configuration register are ignored

8.10.19 TIMER_WDT4_COMMAND_0

CoreWatchdogCommand Register

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	DisableCounter: Write 1 to disable WDT counter if unlocked
0	0x0	StartCounter: Write 1 to enable WDT counter

8.10.20 TIMER_WDT4_UNLOCK_PATTERN_0

CoreWatchdogDisableUnlock Register

Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	UnlockPattern: Must be written with value 0xC45A to allow write to DisableCounter to take effect. The pattern is reset at each write to the Command Register.

8.11 Timer Shared Interrupt Status

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

TimerSrcBitmap is used for Timers 6 through 10 which share a common interrupt line to the controller. Bit b is set if timer 6+b generated an interrupt. The corresponding bit is cleared by writing INTR_CLR bit of the TMR_PCR register.

WatchdogSrcBitmap is used for the 5 watchdog timers. This is the set of Interrupt Status bits from the corresponding CoreWatchdogStatus, but as a bitmap.

8.11.1 SHARED_INTR_STATUS_0

Offset: 0x0 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:6	X	TimerSrcBitmap: Timer[0,9:6] interrupt status in bitmap form
4:0	X	WatchdogSrcBitmap: WDT[4:0] interrupt status in bitmap form

8.11.2 SHARED_TIMER_SECURE_CFG_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx0xxx00000xx0000000000)

Bit	Reset	Description
20	0x0	USEC: TIMERUS_USEC_CFG secure mode config. If this bit is set (=1) then TIMERUS_USEC_CFG register can only be written by secure writes. If this bit is cleared then TIMERUS_USEC_CFG register can be written by secure or non-secure writes. This register can always be read by secure or non-secure reads. Note: this bit should be set if any one of the TMR is enabled to be secure. 1: ENABLE 0: DISABLE

Bit	Reset	Description
16	0x0	WDT4: WDT4 secure mode config. See the WDT0 bit description. This bit is reserved since WDT4 belongs to COP and it has no notion of Secure or Non-Secure transactions
15	0x0	WDT3: WDT3 secure mode config. See the WDT0 bit description.
14	0x0	WDT2: WDT2 secure mode config. See the WDT0 bit description.
13	0x0	WDT1: WDT1 secure mode config. See the WDT0 bit description.
12	0x0	WDT0: WDT0 secure mode config. If this bit is set (=1) then WDT0 registers (i.e. TIMER_WDT0_{CONFIG,COMMAND,UNLOCK_PATTERN}) can only be written by secure writes. A non-secure read will not trigger read-side-effect, but would return proper read value. If this bit is cleared then WDT0 registers can be written by secure or non-secure writes. WDT0 registers can always be read by secure or non-secure reads. Note: if this bit set, then TMR used for this WDT0 should be enabled to be secure. 1: ENABLE 0: DISABLE
9	0x0	TMR9: Timer9 secure mode config. See the TMR0 bit description.
8	0x0	TMR8: Timer8 secure mode config. See the TMR0 bit description.
7	0x0	TMR7: Timer7 secure mode config. See the TMR0 bit description.
6	0x0	TMR6: Timer6 secure mode config. See the TMR0 bit description.
5	0x0	TMR5: Timer5 secure mode config. See the TMR0 bit description.
4	0x0	TMR4: Timer4 secure mode config. See the TMR0 bit description.
3	0x0	TMR3: Timer3 secure mode config. See the TMR0 bit description.
2	0x0	TMR2: Timer2 secure mode config. See the TMR0 bit description.
1	0x0	TMR1: Timer1 secure mode config. See the TMR0 bit description.
0	0x0	TMR0: Timer0 secure mode config. If this bit is set (=1) then TMR0 registers (i.e., TIMER_TMR0_TMR_{PTV,PCR}) can only be written by secure writes. A non-secure read will not trigger read-side-effect, but would return proper read value. If this bit is cleared then TMR0 registers can be written by secure or non-secure writes. TMR0 registers can always be read by secure or non-secure reads. 0: DISABLE 1: ENABLE

9.0 MULTI-PURPOSE I/O PINS AND PIN MULTIPLEXING (PINMUXING)

9.1 Overview

Tegra® 4 devices can be configured with different I/O functions on particular pins to allow use in a variety of different configurations. This section discusses how this is controlled and how the pins themselves are set up.

Many of the pins on Tegra 4 devices are connected to multi-purpose I/O (MPIO) pads. An MPIO can operate in two modes: either acting as a signal for a particular I/O controller, referred to as a Special-Function I/O (SFIO); or as a software-controlled general-purpose I/O function, referred to as GPIO. Each MPIO has up to four SFIO functions as well as being a GPIO.

Though each MPIO has up to 5 functions (a GPIO function and up to 4 SFIO functions), a given MPIO can only act as a single function at a given point in time. The Pinmux controller in Tegra 4 devices includes the logic and registers to select a particular function for each MPIO.

This section describes the following features of MPIOs, the Pinmux controller, and the GPIO controller:

- Basic capabilities of the MPIO pads
- Differences between the variety of MPIO pads
- Mapping of MPIO pads to I/O controllers (i.e., pinmuxing)
- Behavior of the MPIO pads during power-up
- Behavior of the MPIO pads before, during, and after deep sleep
- Recommendations for software programming related to the MPIO pads.

This section covers the Multi-Purpose digital I/O pads. It does not address the special-purpose I/O pads, such as those used for USB, IC_USB, SATA, PCIE, TVO/DAC, MIPI DSI, MIPI CSI, the oscillator, or DRAM interfaces.

9.2 Terms and Acronyms

Some common terms used with Tegra 4 processor pinmuxing are as follows:

- MPIO = Multi-Purpose I/O
- GPIO = General-Purpose I/O
- SFIO = Special-Function I/O
- ST = Standard MPIO pads
- DD = Dual-driver MPIO pads
- OD = Open-drain MPIO pads
- CZ = Controlled-output impedance MPIO pads
- LV = Low-voltage MPIO pads
- Deep sleep

9.3 MPIO Pad Description

Each MPIO pad consists of:

- An output driver with:
  - Tristate capability
  - Drive strength controls AND
  - Push-pull mode, open-drain mode, or both

- An input receiver with:
  - Schmitt mode, CMOS mode, or both
- A weak pull-up and a weak pull-down

Tegra 4 devices include five types of MPIO pads, which all share a common structure. The following table summarizes the differences between the five MPIO pad types.

Table 24: MPIO Pad Types

Pad Type	I/O Rail Voltage	Input Buffer	Output Buffer	I/O Voltage Tolerance	Nominal Pull Strength	"Slew Rate" Control	Drive Strength Control
ST	1.8, 2.8-3.3	Schmitt & CMOS	push-pull	VDDIO	100 kΩ	2 bits, up & down	5 bits, up & down. LPMD
CZ	1.8, 2.8-3.3	Schmitt & CMOS	push-pull	VDDIO	15 kΩ	2 bits, up & down	7 bits, up & down.
DD	1.8, 2.8-3.3	Schmitt & CMOS	push-pull & open-drain	3.3V for open-drain, VDDIO otherwise	50 kΩ	2 bits, up & down	5 bits, up & down. LPMD
LV	1.2, 1.8	CMOS	push-pull	VDDIO	15 kΩ	2 bits, up & down	5 bits, up & down.
OD	1.8, 2.8-3.3	Schmitt & CMOS	open-drain	5V	100 kΩ -- down only	2 bits, down only	5 bits, up. LPMD

The ST (standard) MPIO pads are the most common pads on the chip.

The DD (dual-driver) MPIO pads are similar to the ST pads with the addition of a 3.3V tolerant true open-drain mode. A DD pad can tolerate its I/O pin being pulled up to 3.3V (regardless of supply voltage) as long as the pad's output-driver is set to open-drain mode. There are special power-sequencing considerations when using this functionality.

Note: Refer to Power Sequencing, in *NVIDIA Tegra 4 Series 4-PLUS-1 Quad-Core Processors Data Sheet* (DS-06217-001) for a complete description of the power-up sequencing requirements for Tegra 4 processors.

The OD (open drain) MPIO pads are optimized to tolerate 5V on the I/O pin regardless of the supply voltage. They are similar to ST pads except for an improved I/O voltage tolerance, the absence of a weak pull-up, and the absence of a push-pull output driver.

The CZ (controlled output impedance) MPIO pads are optimized for use in applications requiring a tightly controlled output impedance. They are similar to the ST pads except for changes in the drive strength circuitry and in the weak pull-ups/pull-downs. Tegra 4 processors include CZ pads on the VDDIO_SDMMC1 and VDDIO_SDDMC3 power rails. Each of those rails also includes a pair of CZ_COMP pads. Circuitry within the Tegra 4 device continually matches the output impedance of the CZ pads to the on-board pull-up/-down resistors attached to the CZ_COMP pads.

The LV (low voltage) MPIO pads are optimized for use with a 1.2V supply voltage (and signaling level). They support a 1.8V supply voltage (and signaling level) as a secondary mode. The Tegra 4 processors include LV pads on VDDIO_SDMMC4. The SDMMC4 interface also has a pair of COMP pads, i.e., LV_COMP pads, to generate impedance code.

Note: Refer to Multi-Purpose I/O Pads, in *NVIDIA Tegra 4 Series 4-PLUS-1 Quad-Core Processors Data Sheet* (DS-06217-001) for the list of the pad type, the nominal pull-up/-down strength, and the I/O power rail information associated with each MPIO. See the columns labeled "Pad Type", "Pull Strength", and "Power Rail."

9.4 Pad Controls

The Tegra 4 devices include many controls for each MPIO pad. Some of these controls can be set on a per-pin basis. Other controls are shared across multiple pins.

9.4.1 Per Pad Options

The following controls can be independently configured on a per-pad basis:

PUPD	Internal Pull-up/down option: Option to enable internal Pull-up, Pull-down resistors or neither
TRISTATE	Tristate (high-z) option: Disables or enables the pad's output driver. This setting overrides any other functional setting and also whether pad is selected for SFIO or GPIO. Can be used when the pad direction changes or the pad is assigned to different SFIO to avoid glitches.
E_INPUT	Input Receiver (Enable/Disable): Enables or disables input receiver.
OD	Open Drain option: (Applies to DD pads only) Selects between an open-drain output driver and a push-pull driver.
IO_RESET	Not used.
RCV_SEL	(Applies to OD pads only). Selects between "High VIL/VIH" and "Normal VIL/VIH" receivers. RCV_SEL=1: "High VIL/VIH" RCV_SEL=0: "Normal VIL/VIH"

During normal operation, these per-pad controls are driven by the pinmux controller registers. See the section called "Pinmuxing" below for more information.

During deep sleep, the PMC bypasses and then resets the pinmux controller registers. Software should reprogram these registers as necessary after returning from deep sleep. See the section called "Deep Sleep Behaviors" for more information on the interaction of PMC, software, and the pinmux controller following deep sleep.

9.4.2 Per Pad Control Group

The MPIO pads are partitioned into 35 "pad control groups". The following controls can be configured independently for each pad control group. Pad control groups are associated with a group of pads that share similar functionality for example, GMI, SDMMC.

HSM	High Speed Mode (Enable/Disable)
SCHMT	Schmitt Trigger (Enable/Disable)
LPMD	Low Power Mode. Select Low Power Modes (different impedance/current values)
DRVDN / UP	Drive Down / Up. Driver Output Pull-Up/Pull-Down drive strength code. Normally, the code is 5 bits but for CZ type pads, it is 7 bits.
SLWR/ SLWF	Slew Falling / Rising. Driver Output Pull-Up/Pull-Down slew control code. Normally, the code is 2 bits. In general, the code corresponding to the minimum slew is set (2'b11).

The controls are configured via the "pad control group registers". There is one pad control register per pad control group. During deep sleep, all of these pad control registers automatically return to their power-on-reset state. Software should reprogram these registers as necessary following deep sleep.

Table 25 lists the register address for each pad control group. Additionally, the table describes the bit positions of the controls within each register.

For example, writing a 1 to bit 3 of register 0x700000884 will enable the Schmitt trigger mode for the pads in group "cdev1cfg" pins, CLK1_OUT and CLK1_REQ. Similarly, clearing bits 14 through 23 of register 0x70000900 will minimize the drive strength (both up and down) of pads in the gmacfg pad control group.

Table 25: Pad Control Groups Register Addresses

Pad Control Group	Register Address	HSM	SCHMT	LPMD	DRVDN	DRVUP	SLWR	SLWF
gmacfg	0x70000900	2	3	5:4	18:14	24:20	29:28	31:30
sdio1cfg	0x700008ec	2	3		18:12	26:20	29:28	31:30
sdio3cfg	0x700008b0	2	3		18:12	26:20	29:28	31:30
aocfg0	0x700009ac	2	3	5:4	16:12	24:20	29:28	31:30
aocfg1	0x70000868	2	3	5:4	16:12	24:20	29:28	31:30
aocfg2	0x7000086c	2	3	5:4	16:12	24:20	29:28	31:30
cdev1cfg	0x70000884	2	3	5:4	16:12	24:20	29:28	31:30
cdev2cfg	0x70000888	2	3	5:4	16:12	24:20	29:28	31:30
ceccfg	0x70000938	2	3	5:4	16:12	24:20	29:28	31:30
dap1cfg	0x70000890	2	3	5:4	16:12	24:20	29:28	31:30
dap2cfg	0x70000894	2	3	5:4	16:12	24:20	29:28	31:30
dap3cfg	0x70000898	2	3	5:4	16:12	24:20	29:28	31:30
dap4cfg	0x7000089c	2	3	5:4	16:12	24:20	29:28	31:30
dap5cfg	0x70000998	2	3	5:4	16:12	24:20	29:28	31:30
dbgcfg	0x700008a0	2	3	5:4	16:12	24:20	29:28	31:30
ddccfg	0x700008fc	2	3	5:4	16:12	24:20	29:28	31:30
dev3cfg	0x7000092c	2	3	5:4	16:12	24:20	29:28	31:30
owrcfg	0x70000920	2	3	5:4	16:12	24:20	29:28	31:30
spicfg	0x700008b4	2	3	5:4	16:12	24:20	29:28	31:30
uaacfg	0x700008b8	2	3	5:4	16:12	24:20	29:28	31:30
uabcfg	0x700008bc	2	3	5:4	16:12	24:20	29:28	31:30
uart2cfg	0x700008c0	2	3	5:4	16:12	24:20	29:28	31:30
uart3cfg	0x700008c4	2	3	5:4	16:12	24:20	29:28	31:30
udacfg	0x70000924	2	3	5:4	16:12	24:20	29:28	31:30
atcfg1	0x70000870	2	3	5:4	18:12	26:20	29:28	31:30
atcfg2	0x70000874	2	3	5:4	18:12	26:20	29:28	31:30
atcfg3	0x70000878	2	3	5:4	18:12	26:20	29:28	31:30
atcfg4	0x7000087c	2	3	5:4	18:12	26:20	29:28	31:30
atcfg5	0x70000880	2	3	5:4	18:14	23:19	29:28	31:30
atcfg6	0x70000994	2	3	5:4	18:12	26:20	29:28	31:30
gmecfg	0x70000910	2	3	5:4	18:14	23:19	29:28	31:30
gmfcfg	0x70000914	2	3	5:4	18:14	23:19	29:28	31:30
gmgcfg	0x70000918	2	3	5:4	18:14	23:19	29:28	31:30
gmhcfg	0x7000091c	2	3	5:4	18:14	23:19	29:28	31:30
hvcfg0	0x700009a4	2	3	5:4	16:12	24:20	29:28	31:30

9.4.3 Per I/O Power Rail

These per-power-rail controls are included in the PMC registers which maintain their state during deep sleep. Software does NOT need to reprogram these register following deep sleep. The following table summarizes the functionality of these signals along with how they are controlled in the PMC.

Pin Name	Pin Description	Control/Grouping Mechanism in PMC
E_33V	Active high 3.3V mode select. When low, selects VDDP 1.8V mode.	Generated based on respective PWR_DET signal and the logic is maintained in the PMC, i.e., AO domain By default, it is maintained at Logic 1 to ensure safe power up of I/Os that are pulled to 3.3V
E_NO_IOPWR	Active high. When high, prevents leakage when I/O power is gone while core power is still on.	Maintained per power rails for 16 power rails. Ideally, should have been set during Cold Reset so that I/O rail voltage ramping does not affect the pad. However, it is reset in the chips. Whenever an I/O power rail is shut off on any specific use case, this bit has to be set to Logic1 before the I/O rails are brought down to avoid excessive leakage to the pad.

Nine pads are used for Power Detect status of various I/O rails. They are used to sense the I/O rail voltage level (3.3V or not). They are reflected through the PWR_DET set of registers in the PMC which will in turn control the E_33V pin of the pad.

9.5 Pinmuxing

Tegra 4 processors include many types of I/O controllers (such as I^2C, SDMMC, NAND, and GMI). For some of these controller types, Tegra 4 processors include multiple “controller instances”. For example, Tegra 4 processors include five active I^2C controller instances.

Each controller instance on a Tegra 4 processor communicates with external devices via a set of “external signals”. For each controller instance, each signal can be brought out on (at least) one MPIO. For example, each I^2C controller has two external signals: CLK and DAT. The CLK signal of the I2C1 controller is available on the MPIO whose ball name is GEN1_I2C_SCL.

Many of the pins on Tegra 4 devices are connected to multi-purpose I/O (MPIO) pads. To connect an MPIO to a particular I/O controller, configure it as a Special-Function I/O (SFIO) rather than a General-Purpose I/O (GPIO). Each MPIO has up to four SFIO functions. Each MPIO can as also be programmed to function as a GPIO. For example, the pin named UART3_CTS_can act in one of these five ways:

- As a GPIO
- As the signal CTS for UART3
- As the signal CMD for SDMMC
- As the signal CLK for DTV
- As the signal CS3 for SPI4B controller

Though each MPIO supports multiple functions, a given MPIO can only act as a single function at a given point in time. The Pinmux controller on a Tegra 4 device includes the logic and registers to select a particular function for each MPIO.

Some controller instances have a particular signal available on more than one MPIO. Before using any controller, make sure that the pinmux registers are programmed to bring each signal out on at most ONE MPIO. For example, UART3’s TXD signal is available on the MPIOs whose ball names are ULPI_CLK, GMI_A16, etc. In a system which brings out UART3’s TXD signal on the GMI_A16 ball, the pinmux registers for ULPI_CLK should be programmed to select some other signal.

Some controller instances make their entire set of signals available on two or more sets of MPIOs. That is, such controllers have more than one “interface”. Before using any controller, make sure that the pinmux registers are programmed to bring out the controller’s signals on at most ONE interface.

Note: The Pinmux controller includes one register per MPIO. It can be controlled on a per-pin basis.

Figure 22 shows the pinmux logic associated with a single MPIO. The ENB in the diagram (equivalent to the EN pin at the pad) is active Low. Thus, the pad is active when ENB is Low and is in the High-Z state when ENB is High.

In the figure below, GPIO_OE and SFIO_OE[3:0] are considered active Low; that is, when they are Low, the pad's output is enabled.

Figure 22: Structure of Pinmux Controller (per MPIO)

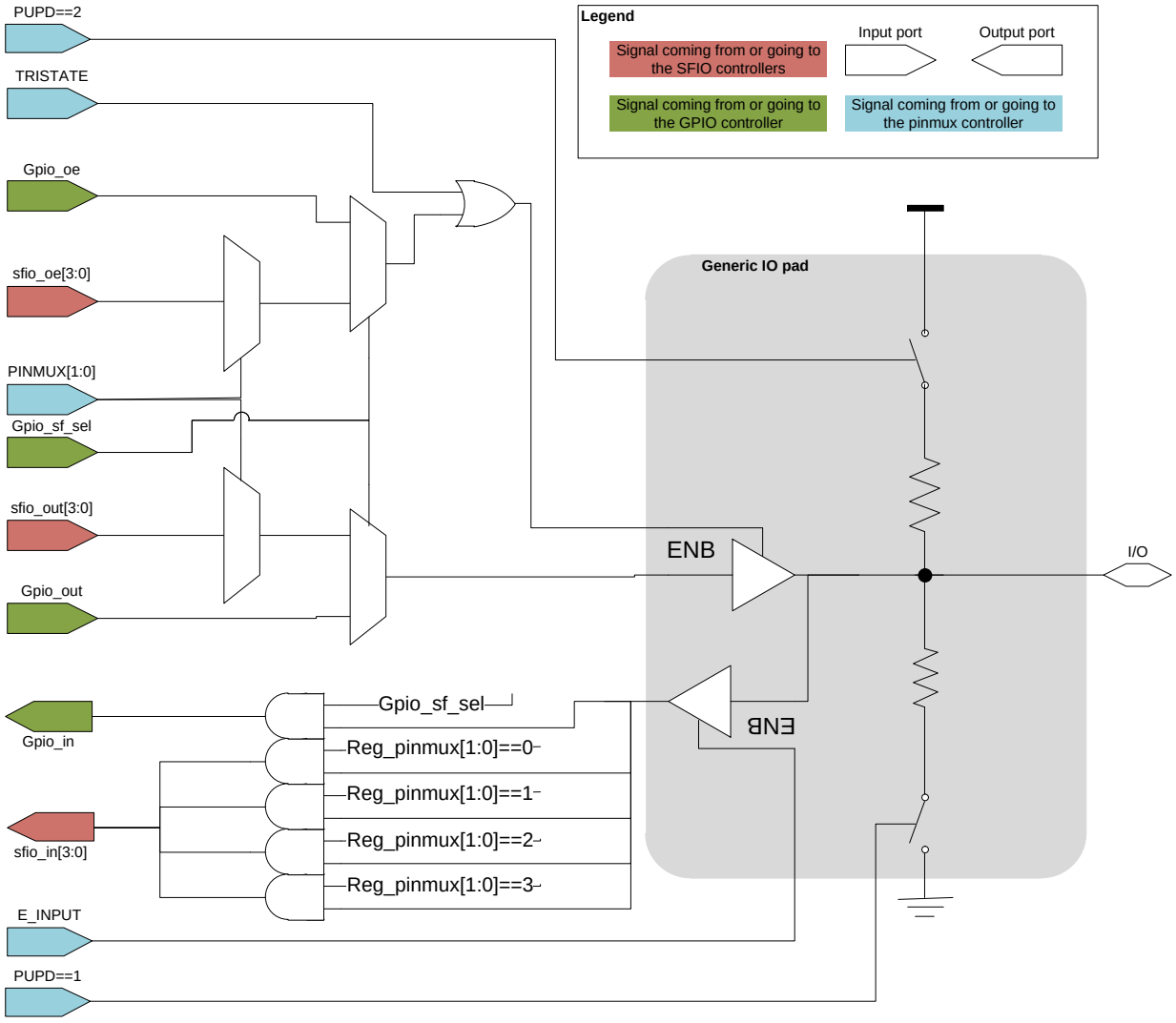


Table 26 describes the layout of each Pinmux controller register.

Table 26: Pinmux Register Format

Field Name	Bit Position	Default Value	Description
RCV_SEL	9	0/1 (depends on the type of interface)	Applicable for "OD" pads
IO_RESET	8	-	This is a dummy pin for MPIO pads and not used. The effect of applying IO_RESET is achieved by keeping A and EN pins of the pads to Logic 0.

Field Name	Bit Position	Default Value	Description
LOCK	7	0	0: Writes to this register are accepted 1: Writes to this register are ignored (until the next wake from Deep Sleep) This is a sticky bit. Once software sets this bit to 1, the only way to clear it is to reset the chip or enter and exit Deep Sleep.
E_OD	6	DD pads:1	0: the pad's output driver operates in push-pull mode. 1: the pad's output driver operates in open-drain mode. WARNING: the DD pads are only 3.3V tolerant when this bit is set. When this bit is cleared, the voltage tolerance is limited to the I/O power supply voltage. Before pulling/driving external I/Os to a value greater than the I/O power supply, this bit must be set to avoid excessive leakage. Default value recommended for DD pads is 1 to ensure that during power up no excessive leakage occurs irrespective of I/O voltage. Weak pull-ups are disabled if E_OD is set to 1 for DD pads.
		Other pads: 0	This bit only matters for DD pads. It is not implemented for other MPIO pad types.
E_INPUT	5	0/1 (depends on the type of functionality)	0: the pad's input receiver is disabled. 1: the pad's input receiver is enabled. Refer to the Tegra 4 Pinmux spreadsheet.
TRISTATE	4	0/1 (depends on the type of functionality)	0: the pad's output driver is enabled. 1: the pad's output driver is disabled (i.e., the output driver is in a high-impedance state). Refer to the Tegra 4 Pinmux spreadsheet.
PUPD	3:2	0/1 (depends on the type of functionality)	0: the pad's weak pull-up and pull-down are both disabled 1: the pad's weak pull-down is enabled and the weak pull-up is disabled. 2: the pad's weak pull-up is enabled and the weak pull-down is disabled. 3: this is an illegal combination. Refer to the Tegra 4 Pinmux spreadsheet.
PINMUX	1:0	00	0: set the pinmux logic for SFIO0. 1: set the pinmux logic for SFIO1. 2: set the pinmux logic for SFIO2. 3: set the pinmux logic for SFIO3. Note that the choice between SFIO and GPIO is controlled via the GPIO registers. Refer to the Tegra 4 Pinmux spreadsheet.

Refer to the Tegra 4 Pinmux spreadsheet for a list of the SFIO functions supported by each MPIO on Tegra 4 devices. Each SFIO is listed in the table with the following format: <ControllerType><ControllerInstance><InterfaceLetter>-<Signal>

- **ControllerType** indicates the type of I/O controller associated with this SFIO function
- **ControllerInstance** indicates the I/O controller, if the Tegra 4 processor has more than one I/O controller of the specified type
- **InterfaceLetter** differentiates between the pin sets, if the controller instance can connect to more than one set of pins
- **Signal** identifies the particular role this SFIO plays for the I/O controller.

For some of the MPIOs, the hardware includes SFIO functions that are omitted from the spreadsheet. Many of these are deprecated. Contact NVIDIA for more information.

9.6 Cold Boot Reset

The following list is a simplified description of the device power on/boot process for Tegra 4 devices concentrating on those aspects which relate to the MPIO pins.

1. System-level hardware executes the power-up sequence. This sequence ends when system-level hardware releases SYS_RESET_N.
2. Each MPIO pin has a deterministic power-on-reset state. The particular reset state for each pin is chosen to minimize the need of on-board components like pull-up resistors in a Tegra 4 based system. For example, the on-chip weak pull-ups are enabled during power-on-reset for pins which are usually used to drive active-low chip selects.
3. Because during power-up the core logic is up before I/O rails, the E_NO_IOPWR control pins of the pads must default to Logic 1.
4. Pins that act as a strap source for cold boot must have their inputs enabled (through E_INPUT control). If they have MPIO Pinmuxing enabled, the strap source should be the primary function.
5. The rising edge of SYS_RESET_N does not trigger any of the MPIOs to change their output state.

> **Note:** Refer to the Multi-Purpose I/O Pads section in *NVIDIA Tegra 4 Series 4-PLUS-1 Quad-Core Processors Data Sheet* (DS-06217-001) for a list of the power-on-reset states for each of the MPIOs. See the column labeled “Power on Reset & Wake/Strap”.

Following the power-up sequence, most of the MPIOs on the chip will stay in their power-on-reset state until system-dependent software (e.g., the Boot Loader or the OS) reprograms them. However, depending on the secondary boot device used in a given system, the Boot ROM may change the state of some of the MPIOs. The following sections list the MPIOs that the Boot ROM uses for each type of secondary boot device.

9.7 Secondary Boot

The following list is a simplified description of the device boot process for Tegra 4 devices concentrating on those aspects which relate to the MPIO pins.

1. System-level hardware executes the power-up sequence. This sequence ends when system-level hardware releases SYS_RESET_N.
2. The Boot ROM on the Tegra 4 device begins executing and programs the on-chip I/O controllers to access the secondary boot device.
3. The Boot ROM on the Tegra 4 device fetches the BCT and Boot Loader from the secondary boot device.
4. If the BCT and Boot Loader are fetched successfully, Boot ROM on the Tegra 4 device yields to the Boot Loader.
5. Otherwise, Boot ROM on the Tegra 4 device enters USB recovery mode.

Each MPIO pin has a deterministic power-on-reset state. The particular reset state for each pin is chosen to minimize the need of on-board components like pull-up resistors in a Tegra 4 based system. For example, the on-chip weak pull-ups are enabled during power-on-reset for pins which are usually used to drive active-low chip selects.

Pins that act as a strap source must have their inputs enabled by default. If they have MPIO functionality, then the default configuration group should choose strap functionality.

> **Note:** Refer to the Multi-Purpose I/O Pads section in *NVIDIA Tegra 4 Series 4-PLUS-1 Quad-Core Processors Data Sheet* (DS-06217-001) for a list of the power-on-reset states for each of the MPIOs. See the column labeled “Power on Reset & Wake/Strap”.

System designs must adhere to the described power-up sequence.

Following the power-up sequence, most MPIOs on the Tegra 4 device will stay in their power-on-reset state until system-dependent software (the Boot Loader or the OS) reprograms them. However, depending on the secondary boot device used in

a given system, the Boot ROM may change the state of some of the MPIOs and associated pinmux so that the Boot ROM can interface with the secondary device to fetch the Boot Loader.

The following subsections list the MPIOs that the Boot ROM uses for each type of secondary boot device. Depending on the type of secondary boot device used, the associated I/O rails have to be brought up by the Boot ROM/PMIC to facilitate the data exchange.

9.7.1 SPI Boot

In systems using a SPI flash as the secondary boot device, the Boot ROM for Tegra 4 devices reconfigures the pinmux for the following pins. SPI4 is used as SFIO-03.

Ball Name	SFIO Usage	I/O Power Rail
GMI_AD5	SPI|4|C|SCK:B	vddio_gmi
GMI_AD6	SPI|4|C|DOUT:B	vddio_gmi
GMI_AD7	SPI|4|C|DIN:B	vddio_gmi
GMI_CS6_N	SPI|4|C|CS1:B	vddio_gmi

9.7.2 eMMC Boot

There are four SDMMC interfaces; the one used as a secondary boot interface is SDMMC4. SDMMC4 has all the necessary functionality for SDMMC assigned as primary function, and thus there is no need to change the Pinmux configuration. However, the Boot ROM has to remove the TRISTATE control bit on these pins (through the Pinmux control register) to ensure the pins do not have the overriding TRISTATE that is making the pins inputs.

9.7.3 NAND Boot

The NAND Interface is multiplexed with the GMI Interface as a non-Primary interface. So the Boot ROM has to configure the Pinmux appropriately to choose these:

Ball Name	SFIO Usage	I/O Power Rail
GMI_WAIT	NAND_BSY0:i	vddio_gmi
GMI_ADV_N	NAND_ALE:o	vddio_gmi
GMI_CLK	NAND_CLE:o	vddio_gmi
GMI_CS4_N	NAND_CE4:o	vddio_gmi
GMI_CS2_N	NAND_CE2:o	vddio_gmi
GMI_CS3_N	NAND_CE3:o	vddio_gmi
GMI_AD0	NAND_D0:b	vddio_gmi
GMI_AD1	NAND_D1:b	vddio_gmi
GMI_AD2	NAND_D2:b	vddio_gmi
GMI_AD3	NAND_D3:b	vddio_gmi
GMI_AD4	NAND_D4:b	vddio_gmi
GMI_AD5	NAND_D5:b	vddio_gmi
GMI_AD6	NAND_D6:b	vddio_gmi
GMI_AD7	NAND_D7:b	vddio_gmi
GMI_AD8	NAND_D8:b	vddio_gmi
GMI_AD9	NAND_D9:b	vddio_gmi
GMI_AD10	NAND_D10:b	vddio_gmi
GMI_AD11	NAND_D11:b	vddio_gmi
GMI_AD12	NAND_D12:b	vddio_gmi
GMI_AD13	NAND_D13:b	vddio_gmi
GMI_AD14	NAND_D14:b	vddio_gmi

Ball Name	SFIO Usage	I/O Power Rail
GMI_AD15	NAND_D15:b	vddio_gmi
NAND_GMI_CLK_LB	nand_gmi_clk_lb:b	vddio_gmi
GMI_WR_N	NAND_WE:o	vddio_gmi
GMI_OE_N	NAND_RE:o	vddio_gmi
GMI_DQS_P	NAND_DQS:b	vddio_gmi
GMI_CS6_N	NAND_CE6:o	vddio_gmi
GMI_CS7_N	NAND_CE7:o	vddio_gmi
GMI_RST_N	NAND_CE5:o	vddio_gmi
GMI_WP_N	NAND_CE5:o	vddio_gmi

9.8 Deep Sleep Behaviors

Deep Sleep is an ultra-low-power standby state in which a Tegra 4 device maintains much of its I/O state while most of the chip is powered off. The following lists offer a simplified description of the deep sleep entry and exit concentrating on those aspects which relate to the MPIO pads.

9.8.1 Deep Sleep Entry

The steps to enter deep sleep are as follows:

1. Software programs the PMC to enter deep sleep.
2. I/O controllers are brought into the IDLE state.
3. The PMC latches much of the Tegra 4 device's I/O state. It is driven by functional logic by sampling the Pinmux controller's output. Latching of the I/O state is triggered by writing into the APBDEV_PMC_DPD_SAMPLE_0 register maintained in the PMC register space.
4. Along with latching the I/O values, various pad controls like E_OD, E_33V, and RCV_SEL have to be latched and driven depending on the pad type.
5. The PMC runs a state machine to drive the control pins in each pad, which puts them in the DPD state. There are two control pins (E_DPD and SEL_DPD) which have to be driven with proper timing as per the Pad datasheet. The PMC logic provides a configurable timer to select this. The timer is provided in the APBDEV_PMC_SEL_DPD_TIM_0 register, and entering into DPD is triggered through APBDEV_PMC_IO_DPD_REQ_0 in the PMC register space.
6. The PMC deasserts CORE_PWR_REQ.
7. The PMIC powers down VDD_CORE.
8. The PMC continues driving the I/O state that it has latched.

> **Note:** For more information about deep sleep entry, refer to the *NVIDIA Tegra 4 Series 4-PLUS-1 Quad-Core Processors Data Sheet* (DS-06217-001).

9.8.2 Deep Sleep Exit

1. The PMC detects a wake event (for example, a rising edge on an appropriately configured MPIO)
2. The PMC asserts CORE_PWR_REQ.
3. The PMIC powers up VDD_CORE.
4. The PMC resets the logic within VDD_CORE.
5. The PMC takes the I/Os out of E_DPD but continues to drive them in SEL_DPD mode.
6. The Boot ROM executes and wakes up software.
7. Software re-initializes the I/O controllers

8. Software changes the Pinmux configurations to the appropriate controller (based on what was configured during LP0) and clears the TRISTATE bit in the Pinmux register. Clearing of the TRISTATE bit is a trigger used for removing SEL_DPD.

> **Note:** For more information about deep sleep exits, refer to the *NVIDIA Tegra 4 Series 4-PLUS-1 Quad-Core Processors Data Sheet* (DS-06217-001).

9.8.3 Pad Capabilities During Deep Sleep

The MPIO pads do not all have identical behavior during deep sleep. They differ with respect to:

- Output buffer behavior during deep sleep:
  - Does it maintain a programmable (0, 1, or tristate) constant value OR
  - Is it capable of changing state while the chip is still in deep sleep (for example, a pin related to the keyboard controller)?
- Input buffer behavior during deep sleep:
  - Is it forcibly disabled OR
  - Can it be enabled for use as a “GPIO wake event” OR
  - Can it be enabled for some other purpose (for example, a “clock request” pin)?
- Weak pull-up/pull-down behavior during deep sleep
  - Are they forcibly disabled OR
  - Can they be configured?
- Behavior coming out of deep sleep
  - All the pads should maintain the state until software configures the I/O controller
- Pads that do not enter DPD mode during LP0
  - Some pads whose outputs are dynamic have a special type and they don’t enter DPD mode during LP0. The PMC has a configurable option to exclude specific pads from entering DPD during LP0 (see the PMC register APBDEV_PMC_DPD_PADS_ORIDE_0)
  - Pads that are associated with PMIC logic do not enter DPD mode during LP0
  - Some pads like JTAG do not enter DPD mode.

Refer to the “Multi-Purpose I/O Pads” section in the document entitled *NVIDIA Tegra 4 Series 4-PLUS-1 Quad-Core Processors Data Sheet* (DS-06217-001) for a summary of the capabilities of each MPIO during deep sleep. See the columns labeled “Output Buffer”, “Input Buffer”, “Weak Pull”, and “After Wake”. The following subsections describe in more detail how to interpret those columns.

9.8.3.1 Output Buffers During Deep Sleep

The output buffers of most MPIO pads are “configurable” during deep sleep. Immediately prior to entering deep sleep, the PMC latches the state of the output buffers for such pads. If the pad was driving high (or low) prior to entering deep sleep, the PMC ensures that it continues to drive high (or low) throughout deep sleep. If the pad was tristated during deep sleep, the PMC ensures that it remains tristated throughout deep sleep.

The weak pull-ups and pull-downs of most of the MPIO pads are “disabled” or “not available” during deep sleep to reduce the Tegra 4 device’s static current consumption. For the remainder of the MPIO pads, the weak pull-up and pull-down are “configurable” during deep sleep. If the pull-up (or pull-down) was enabled prior to entering deep sleep, the PMC ensures that it remains enabled throughout deep sleep. Similarly, if the pull-up (or pull-down) was disabled prior to entering deep sleep, the PMC ensures that it remains enabled throughout deep sleep.

Some MPIO pads have an output buffer which behaves in a “special” way during deep sleep. The output state of these pads may change while the Tegra 4 device remains in deep sleep. For example, the keyboard controller may continue scanning the keypad matrix during deep sleep.

9.8.3.2 Input Buffers During Deep Sleep

The input buffers of most MPIO pads are “disabled” during deep sleep. The input buffers are placed in an ultra-low power state. The Tegra 4 device will not respond to transitions on these pads while it is in deep sleep.

Some MPIO pads have an input buffer which has a “special” behavior during deep sleep. Most of these pads can act as “Deep Sleep Wake Sources”. Others offer some other functional behavior. For example CLK_REQ1 can function as a clock request pin even during deep sleep.

For a complete description of the behavior of these pads during deep sleep, refer to the *NVIDIA Tegra 4 Series 4-PLUS-1 Quad-Core Processors Data Sheet* (DS-06217-001). Pads whose input can be sampled during Deep Sleep are mentioned as ENABLED in the input buffer column. To ensure the PMC wake logic can sense these wake events, the signals must be directly taken from the ZI pin of the pad because pinmux controls are not available during LP0. Pads that are acting as wake source should satisfy the following condition: The pad’s power rail should be active.

9.8.3.3 Pinmuxing Constraints for Special Signals in LP0 Exit

Beyond wake events, there may be specific signals that are used during LP0 exit, that is, signals such as POWERGOOD/OVERCURRENT indication from the PMIC. Because these signals have to be sampled during LP0 recovery, they should be assigned to the primary function of the pinmux so that they can be sampled during LP0 exit (default Pinmux configurations are set at the Primary function).

9.8.3.4 Behavior During Deep Sleep Exit

MPIO pads “hold” their deep sleep state until system-dependent software deasserts the TRISTATE bit in the associated Pinmux controller registers. This gives software the opportunity to re-initialize the pinmux and I/O controllers without any glitches in the state of the MPIO pads.

9.9 GPIO Controller

The GPIO controller for Tegra 4 devices provides the tools for configuring each MPIO for use as a software-controlled GPIO.

The GPIO controller is divided into 8 banks. Each bank handles the GPIO functionality for up to 32 MPIOs. Within a bank, the GPIOs are arranged as four ports of 8 bits each. The ports are labeled consecutively from A through Z and then AA through FF. Ports A through D are in bank 0. Ports E through H are in bank 1. In total, there are 162 GPIOs, and the banking and numbering conventions will have some break in between but will maintain backward compatibility in register configurations for the GPIOs as that of previous generation chips.

> **Note:** Refer to the Multi-Purpose I/O Pads section in *NVIDIA Tegra 4 Series 4-PLUS-1 Quad-Core Processors Data Sheet* (DS-06217-001) for a map of each of the Tegra 4 device MPIO pads to a particular GPIO port and bit. See the column labeled “GPIO.”

Each GPIO can be individually configurable as Output/Input/Interrupt sources with level/edge controls.

GPIO configuration has a lock bit controlling every bit separately. When the LOCK bit is set, the associated control aspects of the bits (for example, whether it is an Output/Input or used as GPIO or SFIO or values driven) cannot be modified (locked). The LOCK bit gets cleared only by system reset; it is sticky. This bit can be used for security-related functionality where an authorized entity owning the GPIO can set the configuration and lock it. The lock bit also covers the GPIO output value, so this may not be varied dynamically once lock is enabled.

The GPIO controller also has masked-write registers. Values written to these registers specify both a mask of bits to be updated in the underlying state (the mask bits are not sticky) as well as new values for that state. Individual bits of the state

can be updated without the need for a read-modify-write sequence. Thus different portions of software can modify the GPIO controller state without coordination.

9.10 Programming Considerations

9.10.1 Controller Instances with Multiple Pin-outs

Several of the I/O controller instances on a Tegra 4 device have more than one pin-out. That is, the controller can communicate with the outside world via more than one set of pins. Special care is warranted when programming the Pinmux controller for these SFIOs. For a given signal on a given controller's interface, only a single pin-out should be selected in the pinmux registers.

For example, TXD is an output signal from UART A. It is available on six pins: ULPI_DATA0, SDMMC1_DATA2, SDMMC3_DATA1, GPIO_PU0, GPIO_PS1, and UART2_RTS_N. Depending on the value programmed into the corresponding pinmux registers, each of those six pins might toggle when UART A transmits data.

Similarly, RXD is an input signal to UART A. It is available on six pins: ULPI_DATA1, SDMMC1_DATA1, SDMMC3_CMD, GPIO_PU1, GPIO_PS2, and UART2_CTS_N. Depending on the value programmed into the corresponding pinmux registers, UART A might see the logical-OR of the signal on those six pins. This scenario will almost certainly lead to data corruption. If the UART A RXD signal is brought out on ULPI_DATA1, the pinmux registers for the other five pins should be programmed to some other function.

9.10.2 Resume From Deep Sleep

As the Tegra 4 device enters Deep Sleep, most of its register state is lost. In particular the pinmux, GPIO, and pad control registers lose their state during LP0. Software is responsible for reprogramming those registers upon resume from deep sleep.

9.10.3 Unused Pins

For each unused MPIO, assert its tristate and disable its input buffer. For pins whose internal pull-up is enabled during power-on-reset, assert the internal pull-up. Otherwise, assert the internal pull-down.

If all of the pins in a pad-control group are unused, set the drive strengths and slew rates to minimum.

If all of the pins on a power-rail are unused, assert E_NOIOPOWER for that rail in the PMC registers.

9.10.4 Security Considerations for the Pinmux Register

Though attacks via Pinmux are mostly Denial Of Service (DoS) attacks, there are specific cases which are a concern, for example:

- Pins interfacing with Platform PMICs (by reprogramming those pinmuxes, one can cause severe user disruption)
- Any I/O device that is transferring secret data (typically like Serial PROMs having keys). By reprogramming the pinmux, it is possible to redirect the data to wrong interface

In such cases, it is better to make use of the LOCK bit in the Pinmux control register so that values cannot be tampered with. The Boot Loader can perform this task so that customer software does not have to configure those pins. As part of LP0 exit, the lock bit will have to be re-enabled, because the lock is not retained across LP0. Similarly, the GPIO controller has a lock bit for each GPIO to preserve the configuration, value driven, and also the pin is configured as GPIO or SFIO. The configuration can be protected using the lock bit in the GPIO register. This way secure software such as a Boot Loader can ensure that certain pins have the required configuration, irrespective of the actions of less secure software.

9.10.5 Drive Strengths

For all pads that do not have corresponding calibration pads, the drive strength has to be programmed. The common default value assumed for the pad drive impedance is 50 ohms and hence pads have to be programmed with a code that can give 50 ohm impedance. Because the code depends on VDDP, i.e., I/O rail, and also PVT (Process, Voltage and Temperature) corners default a code corresponding to TT-NV-NT (Typical process, Normal Voltage, and Normal Temperature) corner as a function of VDDP voltage is chosen.

In Tegra 4 devices, the primary boot device (SDMMC4) has an auto-calibration option. Thus those pads get an impedance code that matches the PVT conditions, etc. and hence enable the boot process. For all other MPIO pads (which are not involved during boot), the BCT can have a value of the code based on the I/O rail and pad data sheet. The same is programmed by the Boot Loader or any platform-specific software, and it overrides the default Power on Reset values. If the values have to be changed based on SI results, then the same can be loaded in the primary device.

9.11 GPIO Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

Each port of each GPIO controller has several registers for the control and monitoring of the port’s 8 GPIO pins. The registers provide per-pin control of the following features:

- GPIO_CNF_* / GPIO_MSK_CNF — Select SPIO or GPIO
- GPIO_OE_* / GPIO_MSK_OE_* — Output Enable
- GPIO_OUT_* / GPIO_MSK_OUT_* — GPIO Output Value
- GPIO_IN_* — GPIO Input Value (Read Only)
- GPIO_INT_STA_* / GPIO_MSK_INT_STA_* — GPIO Interrupt Status
- GPIO_INT_ENB_* / GPIO_MSK_INT_ENB_* — Interrupt Enable
- GPIO_INT_LVL_* / GPIO_MSK_INT_LVL_* — Interrupt Selection (Edge/Level)
- GPIO_INT_CLR_* — Interrupt Flag Set-to-Clear

Many of the control registers are accessible from two offsets. Accesses to the lower offsets affect all 8 pins for the GPIO port. Accesses to the upper offsets provide a per-pin mask capability allowing adjustments to a subset of the pins. Using the upper offsets can eliminate the need for Read-Modify-Write operations. For example, a write to GPIO_MSK_CNF can substitute for a Read-Modify-Write of GPIO_CNF.

Table 27: Tegra 4 GPIO Register Summary

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
GPIO Controller 1 – Port	**A**	**B**	**C**	**D**	**A**	**B**	**C**	**D**
GPIO_CNF_0	000	004	008	00C	080	084	088	08C
GPIO_OE_0	010	014	018	01C	090	094	098	09C
GPIO_OUT_0	020	024	028	02C	0A0	0A4	0A8	0AC
GPIO_IN_0	030	034	038	03C				
GPIO_INT_STA_0	040	044	048	04C	0C0	0C4	0C8	0CC
GPIO_INT_ENB_0	050	054	058	05C	0D0	0D4	0D8	0DC
GPIO_INT_LVL_0	060	064	068	06C	0E0	0E4	0E8	0EC
GPIO_INT_CLR_0	070	074	078	07C				

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
GPIO Controller 2 – Port	**E**	**F**	**G**	**H**	**E**	**F**	**G**	**H**
GPIO_CNF_1	100	104	108	10C	180	184	188	18C
GPIO_OE_1	110	114	118	11C	190	194	198	19C
GPIO_OUT_1	120	124	128	12C	1A0	1A4	1A8	1AC
GPIO_IN_1	130	134	138	13C				
GPIO_INT_STA_1	140	144	148	14C	1C0	1C4	1C8	1CC
GPIO_INT_ENB_1	150	154	158	15C	1D0	1D4	1D8	1DC
GPIO_INT_LVL_1	160	164	168	16C	1E0	1E4	1E8	1EC
GPIO_INT_CLR_1	170	174	178	17C				
GPIO Controller 3 – Port	**I**	**J**	**K**	**L**	**I**	**J**	**K**	**L**
GPIO_CNF_2	200	204	208	20C	280	284	288	28C
GPIO_OE_2	210	214	218	21C	290	294	298	29C
GPIO_OUT_2	220	224	228	22C	2A0	2A4	2A8	2AC
GPIO_IN_2	230	234	238	23C				
GPIO_INT_STA_2	240	244	248	24C	2C0	2C4	2C8	2CC
GPIO_INT_ENB_2	250	254	258	25C	2D0	2D4	2D8	2DC
GPIO_INT_LVL_2	260	264	268	26C	2E0	2E4	2E8	2EC
GPIO_INT_CLR_2	270	274	278	27C				
GPIO Controller 4 – Port	**M**	**N**	**O**	**P**	**M**	**N**	**O**	**P**
GPIO_CNF_3	300	304	308	30C	380	384	388	38C
GPIO_OE_3	310	314	318	31C	390	394	398	39C
GPIO_OUT_3	320	324	328	32C	3A0	3A4	3A8	3AC
GPIO_IN_3	330	334	338	33C				
GPIO_INT_STA_3	340	344	348	34C	3C0	3C4	3C8	3CC
GPIO_INT_ENB_3	350	354	358	35C	3D0	3D4	3D8	3DC
GPIO_INT_LVL_3	360	364	368	36C	3E0	3E4	3E8	3EC
GPIO_INT_CLR_3	370	374	378	37C				
GPIO Controller 5 – Port	**Q**	**R**	**S**	**T**	**Q**	**R**	**S**	**T**
GPIO_CNF_4	400	404	408	40C	480	484	488	48C
GPIO_OE_4	410	414	418	41C	490	494	498	49C
GPIO_OUT_4	420	424	428	42C	4A0	4A4	4A8	4AC
GPIO_IN_4	430	434	438	43C				
GPIO_INT_STA_4	440	444	448	44C	4C0	4C4	4C8	4CC
GPIO_INT_ENB_4	450	454	458	45C	4D0	4D4	4D8	4DC
GPIO_INT_LVL_4	460	464	468	46C	4E0	4E4	4E8	4EC
GPIO_INT_CLR_4	470	474	478	47C				
GPIO Controller 6 – Port	**U**	**V**	**W**	**X**	**U**	**V**	**W**	**X**
GPIO_CNF_5	500	504	508	50C	580	584	588	58C
GPIO_OE_5	510	514	518	51C	590	594	598	59C
GPIO_OUT_5	520	524	528	52C	5A0	5A4	5A8	5AC
GPIO_IN_5	530	534	538	53C				

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
GPIO_INT_STA_5	540	544	548	54C	5C0	5C4	5C8	5CC
GPIO_INT_ENB_5	550	554	558	55C	5D0	5D4	5D8	5DC
GPIO_INT_LVL_5	560	564	568	56C	5E0	5E4	5E8	5EC
GPIO_INT_CLR_5	570	574	578	57C				
GPIO Controller 7 – Port	**Y**	**Z**	**AA**	**BB**	**Y**	**Z**	**AA**	**BB**
GPIO_CNF_6	600	604	608	60C	680	684	688	68C
GPIO_OE_6	610	614	618	61C	690	694	698	69C
GPIO_OUT_6	620	624	628	62C	6A0	6A4	6A8	6AC
GPIO_IN_6	630	634	638	63C				
GPIO_INT_STA_6	640	644	648	64C	6C0	6C4	6C8	6CC
GPIO_INT_ENB_6	650	654	658	65C	6D0	6D4	6D8	6DC
GPIO_INT_LVL_6	660	664	668	66C	6E0	6E4	6E8	6EC
GPIO_INT_CLR_6	670	674	678	67C				
GPIO Controller 8 – Port	**CC**	**DD**	**EE**		**CC**	**DD**	**EE**	
GPIO_CNF_7	700	704	708		780	784	788	
GPIO_OE_7	710	714	718		790	794	798	
GPIO_OUT_7	720	724	728		7A0	7A4	7A8	
GPIO_IN_7	730	734	738					
GPIO_INT_STA_7	740	744	748		7C0	7C4	7C8	
GPIO_INT_ENB_7	750	754	758		7D0	7D4	7D8	
GPIO_INT_LVL_7	760	764	768		7E0	7E4	7E8	
GPIO_INT_CLR_7	770	774	778					

9.11.1 GPIO_CNF_0

Designate whether each pin operates as a GPIO or as an SFIO. By default all pins come up in SFIO mode. These can be programmed to GPIO mode at any stage. This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Configuration Registers

Offset: 0h..fh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15	DISABLE	LOCK_7: Lock access to pin 7 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
14	DISABLE	LOCK_6: Lock access to pin 6 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
13	DISABLE	LOCK_5: Lock access to pin 5 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
12	DISABLE	LOCK_4: Lock access to pin 4 CNF, OE and OUT 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	DISABLE	LOCK_3: Lock access to pin 3 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
10	DISABLE	LOCK_2: Lock access to pin 2 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
9	DISABLE	LOCK_1: Lock access to pin 1 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
8	DISABLE	LOCK_0: Lock access to pin 0 CNF, OE and OUT 0 = DISABLE 1 = ENABLE
7	0x0	BIT_7: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
6	0x0	BIT_6: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
5	0x0	BIT_5: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
4	0x0	BIT_4: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
3	0x0	BIT_3: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
2	0x0	BIT_2: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
1	0x0	BIT_1: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO
0	0x0	BIT_0: Configures each pin to be in either GPIO or SFIO mode 0 = SPIO 1 = GPIO

9.11.2 GPIO_OE_0

GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid.

The set of registers below are used to either drive the signal out or as an Input. This needs to be programmed depending upon whether the pin needs to be in either Input or Output.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Enable Registers

Offset: 10h..1fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = TRI_STATE 1 = DRIVEN

Bit	Reset	Description
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = TRI_STATE 1 = DRIVEN

9.11.3 GPIO_OUT_0

GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be valid. This register will take affect only in GPIO mode. This register is used to drive the value out on a given pin.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Output Value Registers (Read/Write)

Offset: 20h..2fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
6	0x0	BIT_6: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
5	0x0	BIT_5: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
4	0x0	BIT_4: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
3	0x0	BIT_3: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH

Bit	Reset	Description
2	0x0	BIT_2: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
1	0x0	BIT_1: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH
0	0x0	BIT_0: GPIO_CNF.x=1 (in GPIO mode) AND GPIO_OE.x=1 (GPIO output enabled) must be true for this to be a valid state 0 = LOW 1 = HIGH

9.11.4 GPIO_IN_0

GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. This is a read-only register used to read the value from the pin. This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Input Value Registers (Read Only)

Offset: 30h..3fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = LOW 1 = HIGH
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid 0 = LOW 1 = HIGH

All GPIO inputs can be independently programmed to generate an interrupt request.

In addition, the individual trigger level for interrupt on each input pin can be programmed as either active-on-high or active-on-low. For example, to program an active-on-high interrupt on bit 3 of GPIO-PORT_C, write '1' into bit 3 of GPIO_INT.LVL.C register (this sets the interrupt to be active-on-high), and then write '1' into bit 3 of GPIO_INT.ENB.C (this enables interrupt on the named bit).

The interrupt flag status can be read in the appropriate bit of the GPIO_INT.STA.C register. Once the programmed interrupt occurs, status should be cleared by writing into the appropriate bit of the GPIO_INT.CLR.C register. Note that the interrupt

thus generated is routed to the processor only if the corresponding bit for GPIO interrupts in the Secondary interrupt controller is enabled.

9.11.5 GPIO_INT_STA_0

GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. Every GPIO pin generates an Interrupt when switching from Low-High to High-Low. Interrupt status for each port is saved in an Interrupt status register.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Interrupt Status Registers

Offset: 40h..4fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE = ACTIVE
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = IN_ACTIVE 1 = ACTIVE

9.11.6 GPIO_INT_ENB_0

Every bit of the GPIO pin has an enable which, when enabled, routes the Interrupt to the Interrupt controller. This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Interrupt Enable Registers

Offset: 50h..5fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid. 0 = DISABLE 1 = ENABLE

9.11.7 GPIO_INT_LVL_0

The GPIO can detect an interrupt for any edge- or level-sensitive signal.

This is an array of 4 identical register entries; the register fields below apply to each entry

GPIO Port Interrupt Activation Level Registers

Offset: 60h..6fh | Read/Write: R/W | Reset: 0b000000000000000000000000

Bit	Reset	Description
23	0x0	DELTA_7: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
22	0x0	DELTA_6: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
21	0x0	DELTA_5: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
20	0x0	DELTA_4: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
19	0x0	DELTA_3: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
18	0x0	DELTA_2: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
17	0x0	DELTA_1: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
16	0x0	DELTA_0: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE
15	0x0	EDGE_7: 1 means Configure as Edge-Triggered Interrupt
14	0x0	EDGE_6: 1 means Configure as Edge-Triggered Interrupt

Bit	Reset	Description
13	0x0	EDGE_5: 1 means Configure as Edge-Triggered Interrupt
12	0x0	EDGE_4: 1 means Configure as Edge-Triggered Interrupt
11	0x0	EDGE_3: 1 means Configure as Edge-Triggered Interrupt
10	0x0	EDGE_2: 1 means Configure as Edge-Triggered Interrupt
9	0x0	EDGE_1: 1 means Configure as Edge-Triggered Interrupt
8	0x0	EDGE_0: 1 means Configure as Edge-Triggered Interrupt
7	0x0	BIT_7: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
6	0x0	BIT_6: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
5	0x0	BIT_5: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
4	0x0	BIT_4: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
3	0x0	BIT_3: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
2	0x0	BIT_2: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
1	0x0	BIT_1: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH
0	0x0	BIT_0: Interrupt Activation Level or Edge (HIGH for High level or Rising Edge) 0 = LOW 1 = HIGH

9.11.8 GPIO_INT_CLR_0

This write-only register clears the Interrupts that are set. This is valid only in GPIO mode when GPIO_INT.ENB is set.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Interrupt Flag Set-to-Clear Registers

Offset: 70h..7fh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	BIT_7: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
6	0x0	BIT_6: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR

Bit	Reset	Description
5	0x0	BIT_5: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
4	0x0	BIT_4: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
3	0x0	BIT_3: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
2	0x0	BIT_2: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
1	0x0	BIT_1: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR
0	0x0	BIT_0: GPIO mode (GPIO_CNF.x=1) and GPIO_INT.ENB.x=1 must be true for this condition to be valid. 0 = SET 1 = CLEAR

9.11.9 GPIO_MSK_CNF_0

Each register is provided with an individual 16-bit version for enabling Masked Writes to avoid a Read-Modify-Write operation by the firmware. The exception is for the interrupt clear register, whose functionality is combined in the interrupt status register. Individual pins only can be programmed by suitably enabling the write masks in the upper byte of these 16-bit registers.

This is an array of 4 identical register entries; the register fields below apply to each entry.

MASKED PRIMARY GPIO/SFIO Config Registers (Masked Writes)

Offset: 80h..8fh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = SPIO 1 = GPIO
6	RW	0x0	BIT_6: 0 = SPIO 1= GPIO
5	RW	0x0	BIT_5: 0 = SPIO 1 = GPIO
4	RW	0x0	BIT_4: 0 = SPIO 1 = GPIO
3	RW	0x0	BIT_3: 0 = SPIO 1 = GPIO
2	RW	0x0	BIT_2: 0 = SPIO 1 = GPIO
1	RW	0x0	BIT_1: 0 = SPIO 1 = GPIO
0	RW	0x0	BIT_0: 0 = SPIO 1 = GPIO

9.11.10 GPIO_MSK_OE_0

This is an array of 4 identical register entries; the register fields below apply to each entry

GPIO Masked Output Enable (Masked Writes)

Offset: 90h..9fh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = TRI_STATE 1 = DRIVEN
6	RW	0x0	BIT_6: 0 = TRI_STATE 1 = DRIVEN
5	RW	0x0	BIT_5: 0 = TRI_STATE 1 = DRIVEN
4	RW	0x0	BIT_4: 0 = TRI_STATE 1 = DRIVEN
3	RW	0x0	BIT_3: 0 = TRI_STATE 1 = DRIVEN
2	RW	0x0	BIT_2: 0 = TRI_STATE 1 = DRIVEN
1	RW	0x0	BIT_1: 0 = TRI_STATE 1 = DRIVEN
0	RW	0x0	BIT_0: 0 = TRI_STATE 1 = DRIVEN

9.11.11 GPIO_MSK_OUT_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO A-D Masked Output Enable (Masked Writes)

Offset: a0h..afh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = LOW 1 = HIGH
6	RW	0x0	BIT_6: 0 = LOW 1 = HIGH
5	RW	0x0	BIT_5: 0 = LOW 1 = HIGH
4	RW	0x0	BIT_4: 0 = LOW 1 = HIGH
3	RW	0x0	BIT_3: 0 = LOW 1 = HIGH
2	RW	0x0	BIT_2: 0 = LOW 1 = HIGH
1	RW	0x0	BIT_1: 0 = LOW 1 = HIGH
0	RW	0x0	BIT_0: 0 = LOW 1 = HIGH

9.11.12 GPIO_MSK_INT_STA_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO A-D Masked Interrupt Status (Masked Clears)

Offset: c0h..cfh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = IN_ACTIVE 1 = ACTIVE
6	RW	0x0	BIT_6: 0 = IN_ACTIVE 1 = ACTIVE
5	RW	0x0	BIT_5: 0 = IN_ACTIVE 1 = ACTIVE
4	RW	0x0	BIT_4: 0 = IN_ACTIVE 1 = ACTIVE
3	RW	0x0	BIT_3: 0 = IN_ACTIVE 1 = ACTIVE
2	RW	0x0	BIT_2: 0 = IN_ACTIVE 1 = ACTIVE
1	RW	0x0	BIT_1: 0 = IN_ACTIVE 1 = ACTIVE
0	RW	0x0	BIT_0: 0 = IN_ACTIVE 1 = ACTIVE

9.11.13 GPIO_MSK_INT_ENB_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO A-D Masked Interrupt Enable (Masked Writes)

Offset: d0h..dfh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
14	WO	0x0	MSK6: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 0 = DISABLE 1 = ENABLE
7	RW	0x0	BIT_7: 0 = DISABLE 1 = ENABLE
6	RW	0x0	BIT_6: 0 = DISABLE 1 = ENABLE
5	RW	0x0	BIT_5: 0 = DISABLE 1 = ENABLE
4	RW	0x0	BIT_4: 0 = DISABLE 1 = ENABLE
3	RW	0x0	BIT_3: 0 = DISABLE 1 = ENABLE
2	RW	0x0	BIT_2: 0 = DISABLE 1 = ENABLE
1	RW	0x0	BIT_1: 0 = DISABLE 1 = ENABLE
0	RW	0x0	BIT_0: 0 = DISABLE 1 = ENABLE

9.11.14 GPIO_MSK_INT_LVL_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Masked Write Interrupt Activation Levels

Offset: e0h..efh | Read/Write: R/W | Reset: 0b0000000000000000

Bit	R/W	Reset	Description
15	WO	0x0	MSK7: 0=Disable bit for write 1 = ENABLE
14	WO	0x0	MSK6: 0=Disable bit for write 1 = ENABLE
13	WO	0x0	MSK5: 0=Disable bit for write 1 = ENABLE
12	WO	0x0	MSK4: 0=Disable bit for write 1 = ENABLE
11	WO	0x0	MSK3: 0=Disable bit for write 1 = ENABLE
10	WO	0x0	MSK2: 0=Disable bit for write 1 = ENABLE
9	WO	0x0	MSK1: 0=Disable bit for write 1 = ENABLE
8	WO	0x0	MSK0: 0=Disable bit for write 1 = ENABLE
7	RW	0x0	BIT_7: 0 = LOW 1 = HIGH
6	RW	0x0	BIT_6: 0 = LOW 1 = HIGH
5	RW	0x0	BIT_5: 0 = LOW 1 = HIGH
4	RW	0x0	BIT_4: 0 = LOW 1 = HIGH
3	RW	0x0	BIT_3: 0 = LOW 1 = HIGH
2	RW	0x0	BIT_2: 0 = LOW 1 = HIGH
1	RW	0x0	BIT_1: 0 = LOW 1 = HIGH
0	RW	0x0	BIT_0: 0 = LOW 1 = HIGH

9.11.15 GPIO_LOCK_0

Lock bits are used to control the access to CNF and OE registers. When set, no one can write to the CNF and OE bits. These can be programmed ONLY during Boot and get reset by Chip Reset only.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port A - D Lock Bits (Read/Write)

Offset: f0h..ffh | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	DISABLE	BIT_7: Lock access to each pin 0 = DISABLE 1 = ENABLE
6	DISABLE	BIT_6: Lock access to each pin 0 = DISABLE 1 = ENABLE
5	DISABLE	BIT_5: Lock access to each pin 0 = DISABLE 1 = ENABLE
4	DISABLE	BIT_4: Lock access to each pin 0 = DISABLE 1 = ENABLE
3	DISABLE	BIT_3: Lock access to each pin 0 = DISABLE 1 = ENABLE
2	DISABLE	BIT_2: Lock access to each pin 0 = DISABLE 1 = ENABLE
1	DISABLE	BIT_1: Lock access to each pin 0 = DISABLE 1 = ENABLE
0	DISABLE	BIT_0: Lock access to each pin 0 = DISABLE 1 = ENABLE

9.12 Pinmux Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

This section defines the Pinmux selects, Pullup, Pulldn, and E_input programmability for each pin/ball in Tegra 4 devices. Every register has 7 bits defined for Functional Pin Select. The rest are specific to a few pads like OD and IO_RESET.

1:0 PM --> Functional Pin Select.

3:2 PUPD --> Pullup / PullDn control.

4 Tristate --> Tristate Enable / Disable.

5 E_INPUT --> Input Receiver Enable/Disable.

7 LOCK --> Locks access to all configurable bits for pins, including itself.

--> 1 = Enable (preferably done during boot)

--> 0 = Disable (can only be done by full system reset)

8 IO_RESET --> Forces output drivers disabled (PU/PD control only). Required to guarantee pin state on *LPDDR2* pads during POR.

--> IO_RESET_N will override the enable signal of main driver. After power-up, core controls should be enabled first to drive the pad with the same value as programmed by E_PULL. Then IO_RESET_N / E_PULL can be deasserted.

--> If IO_RESET_N is deasserted before core controls come up, there could be unknown outputs or glitches.

Optional bits:

6 OD --> Open Drain Enable / Disable. Specifically for I2C pads.

9 RCV_SEL --> Select between High and Normal VIL/VIH receivers. RCVR_SEL=1: High VIL/VIH RCVR_SEL=0: Normal VIL/VIH

9.12.1 PINMUX_AUX_ULPI_DATA0_0

Offset: 0x3000 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI3	PM: 0 = SPI3 1 = HSI 2 = UARTA 3 = ULPI

9.12.2 PINMUX_AUX_ULPI_DATA1_0

Offset: 0x3004 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI3	PM: 0 = SPI3 1 = HSI 2 = UARTA 3 = ULPI

9.12.3 PINMUX_AUX_ULPI_DATA2_0

Offset: 0x3008 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI3	PM: 0 = SPI3 1 = HSI 2 = UARTA 3 = ULPI

9.12.4 PINMUX_AUX_ULPI_DATA3_0

Offset: 0x300c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI3	PM: 0 = SPI3 1 = HSI 2 = UARTA 3 = ULPI

9.12.5 PINMUX_AUX_ULPI_DATA4_0

Offset: 0x3010 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI2	PM: 0 = SPI2 1 = HSI 2 = UARTA 3 = ULPI

9.12.6 PINMUX_AUX_ULPI_DATA5_0

Offset: 0x3014 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI2	PM: 0 = SPI2 1 = HSI 2 = UARTA 3 = ULPI

9.12.7 PINMUX_AUX_ULPI_DATA6_0

Offset: 0x3018 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	SPI2	PM: 0 = SPI2 1 = HSI 2 = UARTA 3 = ULPI

9.12.8 PINMUX_AUX_ULPI_DATA7_0

Offset: 0x301c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI2	PM: 0 = SPI2 1 = HSI 2 = UARTA 3 = ULPI

9.12.9 PINMUX_AUX_ULPI_CLK_0

Offset: 0x3020 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI1 1 = SPI5 2 = UARTD 3 = ULPI

9.12.10 PINMUX_AUX_ULPI_DIR_0

Offset: 0x3024 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI1 1 = SPI5 2 = UARTD 3 = ULPI

9.12.11 PINMUX_AUX_ULPI_NXT_0

Offset: 0x3028 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI1 1 = SPI5 2 = UARTD 3 = ULPI

9.12.12 PINMUX_AUX_ULPI_STP_0

Offset: 0x302c | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI1 1 = SPI5 2 = UARTD 3 = ULPI

9.12.13 PINMUX_AUX_DAP3_FS_0

Offset: 0x3030 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S2	PM: 0 = I2S2 1 = SPI5 2 = DISPLAYA 3 = DISPLAYB

9.12.14 PINMUX_AUX_DAP3_DIN_0

Offset: 0x3034 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S2	PM: 0 = I2S2 1 = SPI5

Bit	Reset	Description
		2 = DISPLAYA 3 = DISPLAYB

9.12.15 PINMUX_AUX_DAP3_DOUT_0

Offset: 0x3038 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S2	PM: 0 = I2S2 1 = SPI5 2 = DISPLAYA 3 = DISPLAYB

9.12.16 PINMUX_AUX_DAP3_SCLK_0

Offset: 0x303c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S2	PM: 0 = I2S2 1 = SPI5 2 = DISPLAYA 3 = DISPLAYB

9.12.17 PINMUX_AUX_GPIO_PV0_0

Offset: 0x3040 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	USB	PM: 0 = USB 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.18 PINMUX_AUX_GPIO_PV1_0

Offset: 0x3044 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.19 PINMUX_AUX_SDMMC1_CLK_0

Offset: 0x3048 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = CLK12 2 = RSVD2 3 = RSVD3

9.12.20 PINMUX_AUX_SDMMC1_CMD_0

Offset: 0x304c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = SPDIF 2 = SPI4 3 = UARTA

9.12.21 PINMUX_AUX_SDMMC1_DAT3_0

Offset: 0x3050 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = SPDIF

Bit	Reset	Description
		2 = SPI4 3 = UARTA

9.12.22 PINMUX_AUX_SDMMC1_DAT2_0

Offset: 0x3054 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = PWM0 2 = SPI4 3 = UARTA

9.12.23 PINMUX_AUX_SDMMC1_DAT1_0

Offset: 0x3058 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = PWM1 2 = SPI4 3 = UARTA

9.12.24 PINMUX_AUX_SDMMC1_DAT0_0

Offset: 0x305c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = RSVD1 2 = SPI4 3 = UARTA

9.12.25 PINMUX_AUX_CLK2_OUT_0

Offset: 0x3068 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	EXTPERIPH2	PM: 0 = EXTPERIPH2 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.26 PINMUX_AUX_CLK2_REQ_0

Offset: 0x306c | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DAP	PM: 0 = DAP 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.27 PINMUX_AUX_HDMI_INT_0

Offset: 0x3110 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxx0x0x110100)

Bit	Reset	Description
9	0x0	RCV_SEL: 0 = NORMAL 1 = HIGH
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.28 PINMUX_AUX_DDC_SCL_0

Offset: 0x3114 | Read/Write: R/W | Reset: 0x00000230 (0bxxxxxxxxxxxxxxxxxxxxxxx1x0x110000)

Bit	Reset	Description
9	0x1	RCV_SEL: 0 = NORMAL 1 = HIGH
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL

Bit	Reset	Description
		1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C4	PM: 0 = I2C4 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.29 PINMUX_AUX_DDC_SDA_0

Offset: 0x3118 | Read/Write: R/W | Reset: 0x00000230 (0bxxxxxxxxxxxxxxxxxxxxxxxx1x0x110000)

Bit	Reset	Description
9	0x1	RCV_SEL: 0 = NORMAL 1 = HIGH
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C4	PM: 0 = I2C4 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.30 PINMUX_AUX_UART2_RXD_0

Offset: 0x3164 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	IRDA	PM: 0 = IRDA 1 = SPDIF 2 = UARTA 3 = SPI4

9.12.31 PINMUX_AUX_UART2_TXD_0

Offset: 0x3168 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	IRDA	PM: 0 = IRDA 1 = SPDIF 2 = UARTA 3 = SPI4

9.12.32 PINMUX_AUX_UART2_RTS_N_0

Offset: 0x316c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTA	PM: 0 = UARTA 1 = UARTB 2 = RSVD2 3 = SPI4

9.12.33 PINMUX_AUX_UART2_CTS_N_0

Offset: 0x3170 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTA	PM: 0 = UARTA 1 = UARTB 2 = RSVD2 3 = SPI4

9.12.34 PINMUX_AUX_UART3_TXD_0

Offset: 0x3174 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTC	PM: 0 = UARTC 1 = RSVD1 2 = RSVD2 3 = SPI4

9.12.35 PINMUX_AUX_UART3_RXD_0

Offset: 0x3178 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTC	PM: 0 = UARTC 1 = RSVD1

Bit	Reset	Description
		2 = RSVD2 3 = SPI4

9.12.36 PINMUX_AUX_UART3_CTS_N_0

Offset: 0x317c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTC	PM: 0 = UARTC 1 = SDMMC1 2 = DTV 3 = SPI4

9.12.37 PINMUX_AUX_UART3_RTS_N_0

Offset: 0x3180 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	UARTC	PM: 0 = UARTC 1 = PWM0 2 = DTV 3 = DISPLAYA

9.12.38 PINMUX_AUX_GPIO_PU0_0

Offset: 0x3184 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	OWR	PM: 0 = OWR 1 = UARTA 2 = RSVD2 3 = RSVD3

9.12.39 PINMUX_AUX_GPIO_PU1_0

Offset: 0x3188 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = UARTA 2 = RSVD2 3 = RSVD3

9.12.40 PINMUX_AUX_GPIO_PU2_0

Offset: 0x318c | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD0	PM: 0 = RSVD0 1 = UARTA 2 = RSVD2 3 = RSVD3

9.12.41 PINMUX_AUX_GPIO_PU3_0

Offset: 0x3190 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PWM0	PM: 0 = PWM0 1 = UARTA 2 = DISPLAYA 3 = DISPLAYB

9.12.42 PINMUX_AUX_GPIO_PU4_0

Offset: 0x3194 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PWM1	PM: 0 = PWM1 1 = UARTA

Bit	Reset	Description
		2 = DISPLAYA 3 = DISPLAYB

9.12.43 PINMUX_AUX_GPIO_PU5_0

Offset: 0x3198 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PWM2	PM: 0 = PWM2 1 = UARTA 2 = DISPLAYA 3 = DISPLAYB

9.12.44 PINMUX_AUX_GPIO_PU6_0

Offset: 0x319c | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PWM3	PM: 0 = PWM3 1 = UARTA 2 = USB 3 = DISPLAYB

9.12.45 PINMUX_AUX_GEN1_I2C_SDA_0

Offset: 0x31a0 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C1	PM: 0 = I2C1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.46 PINMUX_AUX_GEN1_I2C_SCL_0

Offset: 0x31a4 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C1	PM: 0 = I2C1 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.47 PINMUX_AUX_DAP4_FS_0

Offset: 0x31a8 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S3	PM: 0 = I2S3 1 = RSVD1 2 = DTV 3 = RSVD3

9.12.48 PINMUX_AUX_DAP4_DIN_0

Offset: 0x31ac | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S3	PM: 0 = I2S3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.49 PINMUX_AUX_DAP4_DOUT_0

Offset: 0x31b0 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S3	PM: 0 = I2S3 1 = RSVD1 2 = DTV 3 = RSVD3

9.12.50 PINMUX_AUX_DAP4_SCLK_0

Offset: 0x31b4 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S3	PM: 0 = I2S3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.51 PINMUX_AUX_CLK3_OUT_0

Offset: 0x31b8 | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	EXTPERIPH3	PM: 0 = EXTPERIPH3 1 = RSVD1

Bit	Reset	Description
		2 = RSVD2 3 = RSVD3

9.12.52 PINMUX_AUX_CLK3_REQ_0

Offset: 0x31bc | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DEV3	PM: 0 = DEV3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.53 PINMUX_AUX_GMI_WP_N_0

Offset: 0x31c0 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND 2 = GMI 3 = GMI_ALT

9.12.54 PINMUX_AUX_GMI_IORDY_0

Offset: 0x31c4 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2 1 = RSVD1 2 = GMI 3 = TRACE

9.12.55 PINMUX_AUX_GMI_WAIT_0

Offset: 0x31c8 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SPI4 1 = NAND 2 = GMI 3 = DTV

9.12.56 PINMUX_AUX_GMI_ADV_N_0

Offset: 0x31cc | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND 2 = GMI 3 = TRACE

9.12.57 PINMUX_AUX_GMI_CLK_0

Offset: 0x31d0 | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2 1 = NAND 2 = GMI 3 = TRACE

9.12.58 PINMUX_AUX_GMI_CS0_N_0

Offset: 0x31d4 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND

Bit	Reset	Description
		2 = GMI 3 = USB

9.12.59 PINMUX_AUX_GMI_CS1_N_0

Offset: 0x31d8 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND 2 = GMI 3 = SOC

9.12.60 PINMUX_AUX_GMI_CS2_N_0

Offset: 0x31dc | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2 1 = NAND 2 = GMI 3 = TRACE

9.12.61 PINMUX_AUX_GMI_CS3_N_0

Offset: 0x31e0 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2 1 = NAND 2 = GMI 3 = GMI_ALT

9.12.62 PINMUX_AUX_GMI_CS4_N_0

Offset: 0x31e4 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = USB 1 = NAND 2 = GMI 3 = TRACE

9.12.63 PINMUX_AUX_GMI_CS6_N_0

Offset: 0x31e8 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = NAND 1 = NAND_ALT 2 = GMI 3 = SPI4

9.12.64 PINMUX_AUX_GMI_CS7_N_0

Offset: 0x31ec | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = NAND 1 = NAND_ALT 2 = GMI 3 = SDMMC2

9.12.65 PINMUX_AUX_GMI_AD0_0

Offset: 0x31f0 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND

Bit	Reset	Description
		2 = GMI 3 = RSVD3

9.12.66 PINMUX_AUX_GMI_AD1_0

Offset: 0x31f4 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND 2 = GMI 3 = RSVD3

9.12.67 PINMUX_AUX_GMI_AD2_0

Offset: 0x31f8 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND 2 = GMI 3 = RSVD3

9.12.68 PINMUX_AUX_GMI_AD3_0

Offset: 0x31fc | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND 2 = GMI 3 = RSVD3

9.12.69 PINMUX_AUX_GMI_AD4_0

Offset: 0x3200 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND 2 = GMI 3 = RSVD3

9.12.70 PINMUX_AUX_GMI_AD5_0

Offset: 0x3204 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND 2 = GMI 3 = SPI4

9.12.71 PINMUX_AUX_GMI_AD6_0

Offset: 0x3208 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND 2 = GMI 3 = SPI4

9.12.72 PINMUX_AUX_GMI_AD7_0

Offset: 0x320c | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND

Bit	Reset	Description
		2 = GMI 3 = SPI4

9.12.73 PINMUX_AUX_GMI_AD8_0

Offset: 0x3210 | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = PWM0 1 = NAND 2 = GMI 3 = DTV

9.12.74 PINMUX_AUX_GMI_AD9_0

Offset: 0x3214 | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = PWM1 1 = NAND 2 = GMI 3 = CLDVFS

9.12.75 PINMUX_AUX_GMI_AD10_0

Offset: 0x3218 | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = PWM2 1 = NAND 2 = GMI 3 = CLDVFS

9.12.76 PINMUX_AUX_GMI_AD11_0

Offset: 0x321c | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = PWM3 1 = NAND 2 = GMI 3 = USB

9.12.77 PINMUX_AUX_GMI_AD12_0

Offset: 0x3220 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP

Bit	Reset	Description
		3 = RSVD
1:0	GMI	PM: 0 = SDMMC2 1 = NAND 2 = GMI 3 = RSVD3

9.12.78 PINMUX_AUX_GMI_AD13_0

Offset: 0x3224 | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2 1 = NAND 2 = GMI 3 = RSVD3

9.12.79 PINMUX_AUX_GMI_AD14_0

Offset: 0x3228 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2 1 = NAND 2 = GMI 3 = DTV

9.12.80 PINMUX_AUX_GMI_AD15_0

Offset: 0x322c | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2 1 = NAND 2 = GMI 3 = DTV

9.12.81 PINMUX_AUX_GMI_A16_0

Offset: 0x3230 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = UARTD 1 = TRACE 2 = GMI 3 = GMI_ALT

9.12.82 PINMUX_AUX_GMI_A17_0

Offset: 0x3234 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = UARTD 1 = RSVD1 2 = GMI 3 = TRACE

9.12.83 PINMUX_AUX_GMI_A18_0

Offset: 0x3238 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = UARTD 1 = RSVD1 2 = GMI 3 = TRACE

9.12.84 PINMUX_AUX_GMI_A19_0

Offset: 0x323c | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = UARTD 1 = SPI4

Bit	Reset	Description
		2 = GMI 3 = TRACE

9.12.85 PINMUX_AUX_GMI_WR_N_0

Offset: 0x3240 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND 2 = GMI 3 = SPI4

9.12.86 PINMUX_AUX_GMI_OE_N_0

Offset: 0x3244 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = RSVD0 1 = NAND 2 = GMI 3 = SOC

9.12.87 PINMUX_AUX_GMI_DQS_P_0

Offset: 0x3248 | Read/Write: R/W | Reset: 0x0000003a (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2 1 = NAND 2 = GMI 3 = TRACE

9.12.88 PINMUX_AUX_GMI_RST_N_0

Offset: 0x324c | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = NAND 1 = NAND_ALT 2 = GMI 3 = RSVD3

9.12.89 PINMUX_AUX_GEN2_I2C_SCL_0

Offset: 0x3250 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE

Bit	Reset	Description
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C2	PM: 0 = I2C2 1 = RSVD1 2 = GMI 3 = RSVD3

9.12.90 PINMUX_AUX_GEN2_I2C_SDA_0

Offset: 0x3254 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2C2	PM: 0 = I2C2 1 = RSVD1 2 = GMI 3 = RSVD3

9.12.91 PINMUX_AUX_SDMMC4_CLK_0

Offset: 0x3258 | Read/Write: R/W | Reset: 0x00000136 (0bxxxxxxxxxxxxxxxxxxxxxxxx10x110110)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	GMI	PM: 0 = SDMMC4 1 = RSVD1 2 = GMI 3 = RSVD3

9.12.92 PINMUX_AUX_SDMMC4_CMD_0

Offset: 0x325c | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = RSVD1 2 = GMI 3 = RSVD3

9.12.93 PINMUX_AUX_SDMMC4_DAT0_0

Offset: 0x3260 | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

9.12.94 PINMUX_AUX_SDMMC4_DAT1_0

Offset: 0x3264 | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

9.12.95 PINMUX_AUX_SDMMC4_DAT2_0

Offset: 0x3268 | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

9.12.96 PINMUX_AUX_SDMMC4_DAT3_0

Offset: 0x326c | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

9.12.97 PINMUX_AUX_SDMMC4_DAT4_0

Offset: 0x3270 | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

9.12.98 PINMUX_AUX_SDMMC4_DAT5_0

Offset: 0x3274 | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

9.12.99 PINMUX_AUX_SDMMC4_DAT6_0

Offset: 0x3278 | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = SPI3 2 = GMI 3 = RSVD3

9.12.100 PINMUX_AUX_SDMMC4_DAT7_0

Offset: 0x327c | Read/Write: R/W | Reset: 0x0000013a (0bxxxxxxxxxxxxxxxxxxxxxxxx10x111010)

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC4 1 = RSVD1 2 = GMI 3 = RSVD3

9.12.101 PINMUX_AUX_CAM_MCLK_0

Offset: 0x3284 | Read/Write: R/W | Reset: 0x00000036 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x110110)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VI_ALT3	PM: 0 = VI 1 = VI_ALT1 2 = VI_ALT3 3 = RSVD3

9.12.102 PINMUX_AUX_GPIO_PCC1_0

Offset: 0x3288 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S4	PM: 0 = I2S4 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.103 PINMUX_AUX_GPIO_PBB0_0

Offset: 0x328c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S4	PM: 0 = I2S4 1 = VI 2 = VI_ALT1 3 = VI_ALT3

9.12.104 PINMUX_AUX_CAM_I2C_SCL_0

Offset: 0x3290 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE

Bit	Reset	Description
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP1	PM: 0 = VGP1 1 = I2C3 2 = RSVD2 3 = RSVD3

9.12.105 PINMUX_AUX_CAM_I2C_SDA_0

Offset: 0x3294 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP2	PM: 0 = VGP2 1 = I2C3 2 = RSVD2 3 = RSVD3

9.12.106 PINMUX_AUX_GPIO_PBB3_0

Offset: 0x3298 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	VGP3	PM: 0 = VGP3 1 = DISPLAYA 2 = DISPLAYB 3 = RSVD3

9.12.107 PINMUX_AUX_GPIO_PBB4_0

Offset: 0x329c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP4	PM: 0 = VGP4 1 = DISPLAYA 2 = DISPLAYB 3 = RSVD3

9.12.108 PINMUX_AUX_GPIO_PBB5_0

Offset: 0x32a0 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP5	PM: 0 = VGP5 1 = DISPLAYA 2 = DISPLAYB 3 = RSVD3

9.12.109 PINMUX_AUX_GPIO_PBB6_0

Offset: 0x32a4 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	VGP6	PM: 0 = VGP6 1 = DISPLAYA 2 = DISPLAYB 3 = RSVD3

9.12.110 PINMUX_AUX_GPIO_PBB7_0

Offset: 0x32a8 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S4	PM: 0 = I2S4 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.111 PINMUX_AUX_GPIO_PCC2_0

Offset: 0x32ac | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S4	PM: 0 = I2S4 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.112 PINMUX_AUX_JTAG_RTCK_0

Offset: 0x32b0 | Read/Write: R/W | Reset: 0x00000028 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x101000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RTCK	PM: 0 = RTCK 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.113 PINMUX_AUX_PWR_I2C_SCL_0

Offset: 0x32b4 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	I2CPWR	PM: 0 = I2CPWR 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.114 PINMUX_AUX_PWR_I2C_SDA_0

Offset: 0x32b8 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2CPWR	PM: 0 = I2CPWR 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.115 PINMUX_AUX_KB_ROW0_0

Offset: 0x32bc | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.116 PINMUX_AUX_KB_ROW1_0

Offset: 0x32c0 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.117 PINMUX_AUX_KB_ROW2_0

Offset: 0x32c4 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.118 PINMUX_AUX_KB_ROW3_0

Offset: 0x32c8 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = DISPLAYA 2 = RSVD2 3 = DISPLAYB

9.12.119 PINMUX_AUX_KB_ROW4_0

Offset: 0x32cc | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = DISPLAYA 2 = SPI2 3 = DISPLAYB

9.12.120 PINMUX_AUX_KB_ROW5_0

Offset: 0x32d0 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	KBC	PM: 0 = KBC 1 = DISPLAYA 2 = SPI2 3 = DISPLAYB

9.12.121 PINMUX_AUX_KB_ROW6_0

Offset: 0x32d4 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = DISPLAYA 2 = DISPLAYA_ALT 3 = DISPLAYB

9.12.122 PINMUX_AUX_KB_ROW7_0

Offset: 0x32d8 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = CLDVFS 3 = UARTA

9.12.123 PINMUX_AUX_KB_ROW8_0

Offset: 0x32dc | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = CLDVFS 3 = UARTA

9.12.124 PINMUX_AUX_KB_ROW9_0

Offset: 0x32e0 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = UARTA

9.12.125 PINMUX_AUX_KB_ROW10_0

Offset: 0x32e4 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = RSVD2 3 = UARTA

9.12.126 PINMUX_AUX_KB_COL0_0

Offset: 0x32fc | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = USB 2 = SPI2 3 = EMC_DLL

9.12.127 PINMUX_AUX_KB_COL1_0

Offset: 0x3300 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SPI2 3 = EMC_DLL

9.12.128 PINMUX_AUX_KB_COL2_0

Offset: 0x3304 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SPI2 3 = RSVD3

9.12.129 PINMUX_AUX_KB_COL3_0

Offset: 0x3308 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = DISPLAYA 2 = PWM2 3 = UARTA

9.12.130 PINMUX_AUX_KB_COL4_0

Offset: 0x330c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = OWR 2 = SDMMC3 3 = UARTA

9.12.131 PINMUX_AUX_KB_COL5_0

Offset: 0x3310 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SDMMC1 3 = RSVD3

9.12.132 PINMUX_AUX_KB_COL6_0

Offset: 0x3314 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SPI2 3 = RSVD3

9.12.133 PINMUX_AUX_KB_COL7_0

Offset: 0x3318 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	KBC	PM: 0 = KBC 1 = RSVD1 2 = SPI2 3 = RSVD3

9.12.134 PINMUX_AUX_CLK_32K_OUT_0

Offset: 0x331c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	BLINK	PM: 0 = BLINK 1 = SOC 2 = RSVD2 3 = RSVD3

9.12.135 PINMUX_AUX_SYS_CLK_REQ_0

Offset: 0x3320 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SYSCLK	PM: 0 = SYSCLK 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.136 PINMUX_AUX_CORE_PWR_REQ_0

Offset: 0x3324 | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PWRON	PM: 0 = PWRON 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.137 PINMUX_AUX_CPU_PWR_REQ_0

Offset: 0x3328 | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	CPU	PM: 0 = CPU 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.138 PINMUX_AUX_PWR_INT_N_0

Offset: 0x332c | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	PMI	PM: 0 = PMI 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.139 PINMUX_AUX_CLK_32K_IN_0

Offset: 0x3330 | Read/Write: R/W | Reset: 0x00000020 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	CLK	PM: 0 = CLK 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.140 PINMUX_AUX_OWR_0

Offset: 0x3334 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxx0x0x110000)

Bit	Reset	Description
9	0x0	RCV_SEL: 0 = NORMAL 1 = HIGH
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	OWR	PM: 0 = OWR 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.141 PINMUX_AUX_DAP1_FS_0

Offset: 0x3338 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	I2S0	PM: 0 = I2S0 1 = HDA 2 = GMI 3 = RSVD3

9.12.142 PINMUX_AUX_DAP1_DIN_0

Offset: 0x333c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S0	PM: 0 = I2S0 1 = HDA 2 = GMI 3 = RSVD3

9.12.143 PINMUX_AUX_DAP1_DOUT_0

Offset: 0x3340 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S0	PM: 0 = I2S0 1 = HDA 2 = GMI 3 = RSVD3

9.12.144 PINMUX_AUX_DAP1_SCLK_0

Offset: 0x3344 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S0	PM: 0 = I2S0 1 = HDA 2 = GMI 3 = RSVD3

9.12.145 PINMUX_AUX_CLK1_REQ_0

Offset: 0x3348 | Read/Write: R/W | Reset: 0x00000024 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x100100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	DAP	PM: 0 = DAP 1 = DAP1 2 = RSVD2 3 = RSVD3

9.12.146 PINMUX_AUX_CLK1_OUT_0

Offset: 0x334c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	EXTPERIPH1	PM: 0 = EXTPERIPH1 1 = DAP2 2 = RSVD2 3 = RSVD3

9.12.147 PINMUX_AUX_SPDIF_IN_0

Offset: 0x3350 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPDIF	PM: 0 = SPDIF 1 = USB 2 = RSVD2 3 = RSVD3

9.12.148 PINMUX_AUX_SPDIF_OUT_0

Offset: 0x3354 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	SPDIF	PM: 0 = SPDIF 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.149 PINMUX_AUX_DAP2_FS_0

Offset: 0x3358 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S1	PM: 0 = I2S1 1 = HDA 2 = RSVD2 3 = RSVD3

9.12.150 PINMUX_AUX_DAP2_DIN_0

Offset: 0x335c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S1	PM: 0 = I2S1 1 = HDA 2 = RSVD2 3 = RSVD3

9.12.151 PINMUX_AUX_DAP2_DOUT_0

Offset: 0x3360 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S1	PM: 0 = I2S1 1 = HDA 2 = RSVD2 3 = RSVD3

9.12.152 PINMUX_AUX_DAP2_SCLK_0

Offset: 0x3364 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	I2S1	PM: 0 = I2S1 1 = HDA 2 = RSVD2 3 = RSVD3

9.12.153 PINMUX_AUX_SPI2_MOSI_0

Offset: 0x3368 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI6	PM: 0 = SPI6 1 = CLDVFS 2 = RSVD2 3 = RSVD3

9.12.154 PINMUX_AUX_SPI2_MISO_0

Offset: 0x336c | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI6	PM: 0 = SPI6 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.155 PINMUX_AUX_SPI2_CS0_N_0

Offset: 0x3370 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	SPI6	PM: 0 = SPI6 1 = SPI1 2 = RSVD2 3 = RSVD3

9.12.156 PINMUX_AUX_SPI2_SCK_0

Offset: 0x3374 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI6	PM: 0 = SPI6 1 = CLDVFS 2 = RSVD2 3 = RSVD3

9.12.157 PINMUX_AUX_SPI1_MOSI_0

Offset: 0x3378 | Read/Write: R/W | Reset: 0x00000035 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x110101)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = RSVD0 1 = SPI1 2 = SPI2 3 = DAP2

9.12.158 PINMUX_AUX_SPI1_SCK_0

Offset: 0x337c | Read/Write: R/W | Reset: 0x00000039 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111001)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = RSVD0 1 = SPI1 2 = SPI2 3 = RSVD3

9.12.159 PINMUX_AUX_SPI1_CS0_N_0

Offset: 0x3380 | Read/Write: R/W | Reset: 0x00000039 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111001)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI6 1 = SPI1 2 = SPI2 3 = RSVD3

9.12.160 PINMUX_AUX_SPI1_MISO_0

Offset: 0x3384 | Read/Write: R/W | Reset: 0x00000035 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110101)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = RSVD0 1 = SPI1 2 = SPI2 3 = RSVD3

9.12.161 PINMUX_AUX_SDMMC3_CLK_0

Offset: 0x3390 | Read/Write: R/W | Reset: 0x00000034 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = SPI3

9.12.162 PINMUX_AUX_SDMMC3_CMD_0

Offset: 0x3394 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	SDMMC3	PM: 0 = SDMMC3 1 = PWM3 2 = UARTA 3 = SPI3

9.12.163 PINMUX_AUX_SDMMC3_DAT0_0

Offset: 0x3398 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = SPI3

9.12.164 PINMUX_AUX_SDMMC3_DAT1_0

Offset: 0x339c | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = PWM2 2 = UARTA 3 = SPI3

9.12.165 PINMUX_AUX_SDMMC3_DAT2_0

Offset: 0x33a0 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = PWM1 2 = DISPLAYA 3 = SPI3

9.12.166 PINMUX_AUX_SDMMC3_DAT3_0

Offset: 0x33a4 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = PWM0 2 = DISPLAYB 3 = SPI3

9.12.167 PINMUX_AUX_HDMI_CEC_0

Offset: 0x33e0 | Read/Write: R/W | Reset: 0x00000070 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	CEC	PM: 0 = CEC 1 = SDMMC3 2 = RSVD2 3 = SOC

9.12.168 PINMUX_AUX_SDMMC1_WP_N_0

Offset: 0x33e4 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC1	PM: 0 = SDMMC1 1 = CLK12 2 = SPI4 3 = UARTA

9.12.169 PINMUX_AUX_SDMMC3_CD_N_0

Offset: 0x33e8 | Read/Write: R/W | Reset: 0x00000038 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
1:0	SDMMC3	PM: 0 = SDMMC3 1 = OWR 2 = RSVD2 3 = RSVD3

9.12.170 PINMUX_AUX_SPI1_CS1_N_0

Offset: 0x33ec | Read/Write: R/W | Reset: 0x00000039 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111001)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RSVD1	PM: 0 = SPI6 1 = RSVD1 2 = SPI2 3 = I2C1

9.12.171 PINMUX_AUX_SPI1_CS2_N_0

Offset: 0x33f0 | Read/Write: R/W | Reset: 0x00000039 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x111001)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SPI1	PM: 0 = SPI6 1 = SPI1 2 = SPI2 3 = I2C1

9.12.172 PINMUX_AUX_USB_VBUS_EN0_0

Offset: 0x33f4 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	USB	PM: 0 = USB 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.173 PINMUX_AUX_USB_VBUS_EN1_0

Offset: 0x33f8 | Read/Write: R/W | Reset: 0x00000074 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01110100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
6	ENABLE	OD: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	USB	PM: 0 = USB 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.174 PINMUX_AUX_SDMMC3_CLK_LB_IN_0

Offset: 0x33fc | Read/Write: R/W | Reset: 0x00000024 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100100)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.175 PINMUX_AUX_SDMMC3_CLK_LB_OUT_0

Offset: 0x3400 | Read/Write: R/W | Reset: 0x00000030 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x110000)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	SDMMC3	PM: 0 = SDMMC3 1 = RSVD1 2 = RSVD2 3 = RSVD3

9.12.176 PINMUX_AUX_NAND_GMI_CLK_LB_0

Offset: 0x3404 | Read/Write: R/W | Reset: 0x00000022 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100010)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = SDMMC2 1 = NAND 2 = GMI 3 = RSVD3

9.12.177 PINMUX_AUX_RESET_OUT_N_0

Offset: 0x3408 | Read/Write: R/W | Reset: 0x00000023 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x100011)

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	RESET_OUT_N	PM: 0 = RSVD0 1 = RSVD1 2 = RSVD2 3 = RESET_OUT_N

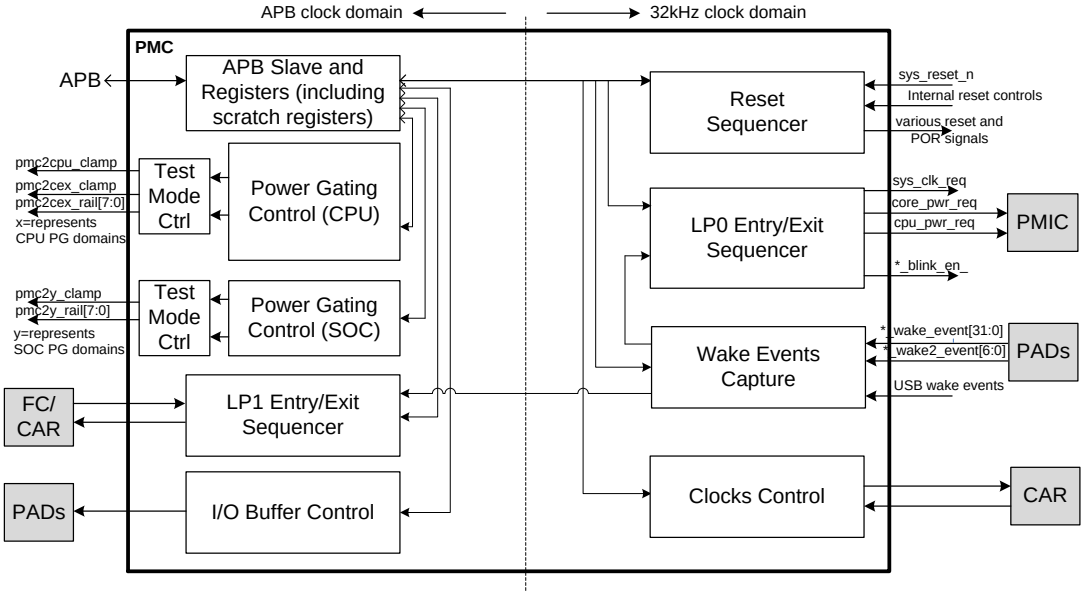
[THIS PAGE INTENTIONALLY LEFT BLANK]

10.0 POWER MANAGEMENT CONTROLLER

The Power Management Controller (PMC) block interacts with an external Power Manager Integrated Circuit (PMIC). The PMC controls the entry and exit of the system from different sleep modes. It provides power-gating controllers for SOC and CPU power partitions and also provides scratch storage to save some of the context during sleep modes (when the CPU and/or SOC power rails are off).

Sleep (LP1) and deep sleep (LP0) require specific logic to maintain some states and control the power domains, including signaling to the external PMIC to provide power to the main logic in the Tegra® 4 devices. All this logic is centralized in the PMC block.

Figure 23: PMC Block Diagram



Glossary

- Cold boot: The SoC partition power transitions from OFF to ON with no previous state available. Software must construct all states from scratch. Boot ROM is executed. DRAM is brought on-line.
- Warm boot: Exit from LP0 state.
- LP0: Low Power 0 state. The system state is saved in the PMC and DRAM, DRAM is put in self-refresh, the PMC is configured to monitor LP0 exit wake events, and the VDD_SOC and VDD CPU rails are powered off. Also called the Deep Sleep state.
- LP1: Low Power 1 state. Devices are power-gated, SoC clock domains are set to the minimum frequency (32 kHz) the flow controller is configured to monitor LP1 exit wake events, DRAM is put in self-refresh, and the VDD_CPU rail is powered off. Also known as Suspend state.
- TSC: ARM Generic Timer System Counter (also known as Time Stamp Counter)
- VDD_AO: Always ON power rail (also known as VDD_RTC power rail)
- VDD_SOC: SOC power rail (also known as VDD_CORE power rail)
- VDD_CPU: CPU power rail

10.1 External Interface

Table 28: PMC External Interface Signals

Signal Name	Type	Signal Description
SYS_RESET_N	In	Hardware active-low reset signal. When this signal is asserted, the entire chip is reset including the PMC.
CORE_PWR_REQ	Out	Used to signal to the external PMIC device when to toggle the VDD_SOC and VDD_CPU voltage rails.
CPU_PWR_REQ	Out	
Wake-up Inputs	In	Up to 38 pins designated as wake-up are used for triggering wake-up from Deep Sleep mode. Note, there 4 additional (internal) wake events from Deep Sleep mode for USB (UTMIP and UHSIC).
SYS_CLK_REQ	Out	For systems where the system clock reference can be disabled during sleep
CLK_32K_OUT	Out	32 kHz clock out (active even during Deep Sleep). Can be used for systems that want a blinking LED during Deep Sleep

10.1.1 Flow Controller Interface

The Tegra 4 flow controller manages CPU power and rail gating transitions. When requested from the flow controller, the PMC CPU power domain sequencers power down CPU complex hardware. See the Flow Controller section for more information on CPU power gating

10.1.2 VDD_CPU Power-Good Signal

Some external PMICs support signaling when the VDD_CPU rail is fully powered on (CPU_PWR_GOOD). When enabled, CPU_PWR_GOOD signals when the PMC should start its internal CPUPWRGOOD_TIMER..

If PMC_CNTRL[CPUPWRGOOD_EN] is enabled:

- The PMC toggles CPU_PWR_REQ to signal the external PMIC to turn on VDD_CPU
- The PMC waits for a rising edge on CPU_PWR_GOOD
- Start the CPUPWRGOOD_TIMER and wait for it to expire
- The PMC sets PMC_PWRGATE_STATUS[CRAIL]
- The PMC ACKs the flow controller to complete the CPU power transition

If PMC_CNTRL[CPUPWRGOOD_EN] is disabled:

- The PMC toggles CPU_PWR_REQ to signal the external PMIC to turn on VDD_CPU
- The PMC starts the CPUPWRGOOD_TIMER and waits for it to expire
- The PMC sets PMC_PWRGATE_STATUS[CRAIL]
- The PMC ACKs the flow controller to complete the CPU power transition

10.1.2.1 Power-Good Software Configuration

- At cold boot:
  - Program CPU_PWR_GOOD pinmux appropriately (see PMC_CNTRL[CPUPWRGOOD_SEL]).
  - Program PMC_CPUPWRGOOD_TIMER.

The timeout register values need to be programmed differently based on whether the PowerGood feature is enabled or disabled. PMC_CPUPWRGOOD_TIMER will be shorter when the Power-Good feature is enabled because timer margin is not needed to account for VDD_CPU power up variances.

- Enable power-good signal usage by writing to PMC_CNTRL[CPUPWRGOOD_EN].

- The Power-Good feature cannot be enabled across LP0 transitions. Prior to LP0 entry:
  - Context save the PMC_CNTRL[CPUPWRGOOD_EN] bit and corresponding CPUPWRGOOD_TIMER value in DRAM.
  - Next update CPUPWRGOOD_TIMER with a value that is appropriate for the PowerGood feature.
  - Clear PMC_CNTRL[CPUPWRGOOD_EN].
- At LP0 exit:
  - Restore the state of PMC_CNTRL[CPUPWRGOOD_EN] followed by the CPUPWRGOOD_TIMER value.

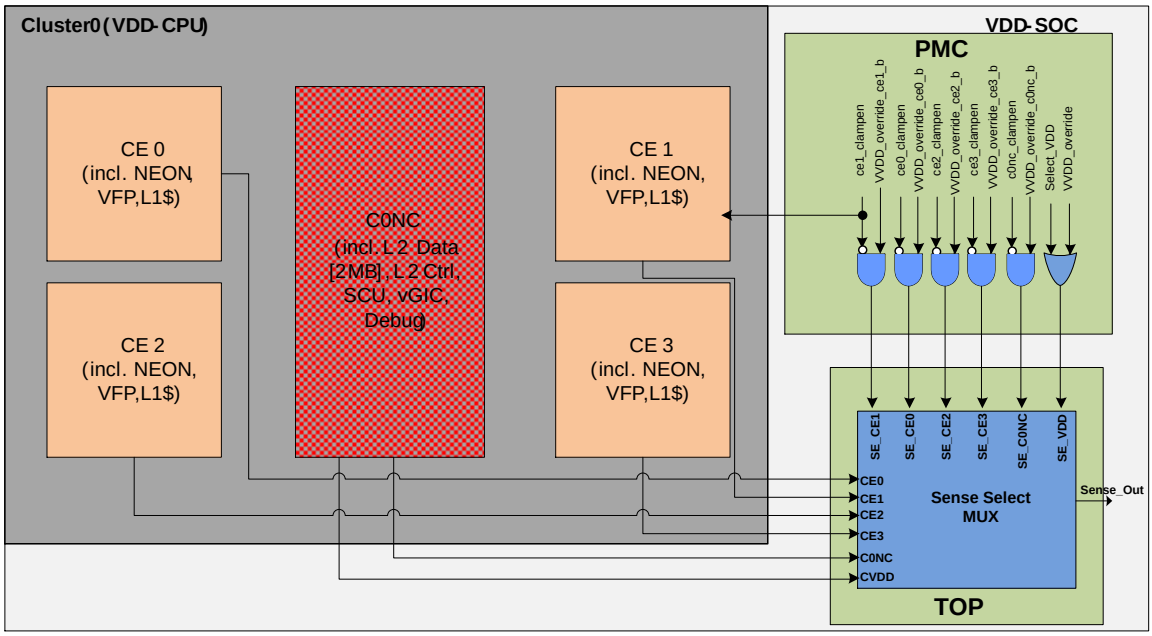
10.2 CPU Voltage Sensing Control

The Voltage Regulator (VR) delivers power to the board, then to the package and to the transistors in the silicon. The VR regulates its voltage output based on the sense feedback. To reduce inaccuracy in the VR output voltage level, the sense voltage needs to be as close to the transistors as possible.

For a power-gated partition, if the sense point is at the real VDD power grid, then the drop across the power-gate adds to the inaccuracy of the sense voltage. To reduce this inaccuracy, virtual VDD can be sensed. However, if the PG partition is power-gated, then its virtual VDD is off and cannot be used for sense feedback. Because partitions are power-gated/ungated dynamically, there needs to be control logic to provide dynamic control for the voltage-sense mux such that the virtual VDD is selected only when the corresponding PG partition is power-ungated.

As shown in the following diagram, a 6-input mux is used to select one (or more) of sensed voltages. The 6-input voltages are VVDD of each of the 5 PG partitions (in cluster0) and the real CPU VDD. The mux is expected to select the VVDD of all PG partitions that are power-ungated (powered on). If none of the 5 PG partitions are power-ungated, then real VDD is selected as the sense voltage.

Figure 24: CPU Voltage Sensing Mux Control Signals



The PMC provides a 6-bit mux select bus. The mux corresponding to 5 partitions is selected if those partitions are powered on. During normal operation, selection of the correct voltage sense signal is handled entirely by the PMC. For debug purposes, a per partition override register (see the CPU_VSENSE_OVERRIDE register) is provided which can be configured to disable input from any of the 5 PG partitions, and select real VDD. The sensing mux select control needs to be as follows:

- Never select the VVDD of a partition which is power-gated
- Select VVDD of all partitions that are power-ungated
- When all partitions are power-gated, select real VDD
  - During transition, both power-ungated (VVDD) and real VDD can be selected.

10.3 Generic Timer's System Counter (TSC)

The ARM Generic Timer Specification requires a system counter (also known as a Time Stamp Counter) to be implemented in SoCs. The Cortex™-A15 timers use this system counter as their reference clock input.

For the Generic Timer Specification, refer to ARM Architectural Reference Manual (ARMv7-A), section "The Generic Timer".

The ARM system counter requirements can be grouped into two groups: basic counter functionality and architectural control and status registers functionality.

10.3.1 Basic Counter Functionality

The basic counter functionality requirements are:

- It should run at a constant clock frequency regardless of the power and clocking state of the processor cores using it.
- A lower bound on the clock frequency of the counter is in the range 1-10MHz.
- When the system is running with an inherently low clock frequency, it is acceptable that the precision of the counter is reduced.
- It must maintain the required counter accuracy, so that it does not gain or lose more than one second in a 24-hour period.
- The size of the counter is recommended to be 64 bits to avoid any roll-over issues. For a counter clock frequency less than 50 MHz, a 56-bit counter is acceptable.

10.3.1.1 Basic Functionality Implementation

Tegra 4 devices include a 56-bit free-running TSC counting at the main oscillator clock frequency, and is reset by pmc-rst (also known as main-rst). During LP0, the oscillator clock can be disabled. The PMC provides support for running the counter off a 32.768 kHz clock.

To save power, in the TSC implementation, only an offset counter runs in 32.768 kHz mode. All external (to PMC) visible views of the TSC (i.e., the TSC bus from the PMC, and TSC register readable value) remain frozen during this mode.

Refer to the TSC Clock Switch Programming Requirement section below for more information.

10.3.1.2 TSC Clock Switch Programming Requirement

Normally, the TSC runs at the oscillator clock, which may be disabled during LP0.

Oscillator to 32.768 kHz Switch

If the oscillator needs to be disabled, then software has to ensure that the TSC clock is switched to 32.768 kHz before the oscillator is disabled. The clock switch (to 32.768 kHz) is triggered when software sets the PMC_DPD_ENABLE[TSC_MULT_EN] bit. Hardware waits for the first rising edge (on 32.768 kHz) before switching the clock. Software needs to use the following steps to ensure the TSC clock is switched (to 32.768 kHz):

- Set the PMC_DPD_ENABLE[TSC_MULT_EN] bit
- Poll for PMC_TSC_MULT[MULT_STS] to be ‘1’ OR wait for 31 µs (=1 cycle of 32.768 kHz)
- Initiate OSC disable

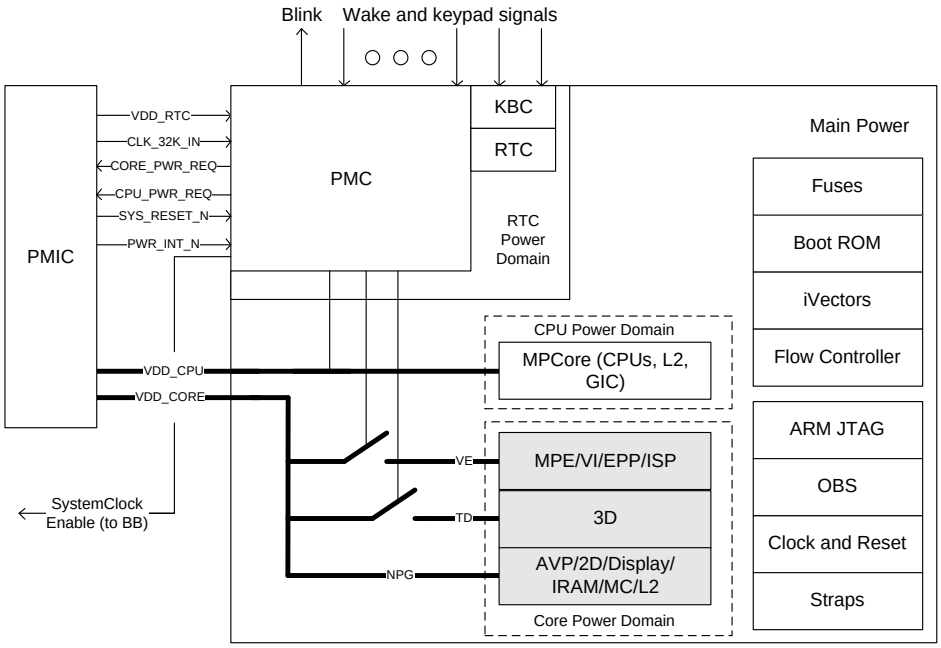
32.768 kHz to Oscillator Switch

Similar to the oscillator-to-32.768 kHz switch, hardware waits for the first rising edge on 32.768 kHz, before switching to the oscillator clock. It must be ensured that OSC is enabled before the TSC clock is switched to the oscillator. The PMC ensures that the oscillator is enabled at the LP0 exit. The LP0 restore software has to switch the TSC clock to the oscillator by clearing the PMC_DPD_ENABLE[TSC_MULT_EN] bit. After switching to the oscillator clock, software does not have to wait for PMC_TSC_MULT[MULT_STS] to be ‘0’ because 32.768 kHz is always running.

10.4 Functionality

The PMC is a part of the RTC power domain. The PMC manages the interface with the external PMIC, including the hardware reset pin, the 32.768 kHz clock, and the power request signal. A schematic view of the power domains inside the Tegra 4 devices is shown below.

Figure 25: Power Domains in Tegra 4 Devices



An important aspect of the PMC is the wake-up and reset logic. The following subsections show the expected behavior of the PMC under three scenarios:

- Frozen boot, that is, the VDD_RTC transitioning from OFF to ON, with or without VDD_CORE and VDD_CPU also transitioning from OFF to ON
- Deep Sleep wake-up, where the PMC is responsible to track wake-up events and to request a transition of VDD_CORE and optionally VDD_CPU from OFF to ON
- Suspend mode wake-up, where the PMC is responsible for establishing the power back to the CPU partition

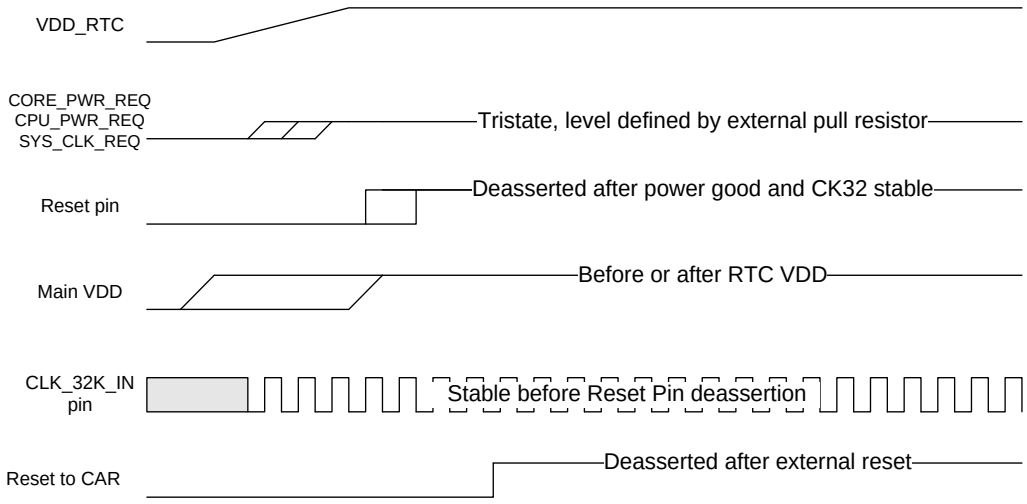
When reset is deasserted to the Tegra 4 devices, the PMC logic performs minimal processing. It forwards a reset signal and the 32.768 kHz clock to the Clock and Reset (CAR) block.

10.4.1 Frozen Boot Sequence

The Frozen Boot wake-up sequence happens when power has been deasserted to the RTC domain:

1. The PMIC detects a turn-on condition and enables power to VDD_RTC.
2. Simultaneously to VDD_RTC enable or shortly afterwards, the PMIC enables power to VDD_CORE and optionally to VDD_CPU.
3. More than 1 ms later, the PMIC enables power to 1.8V I/O rails that need to be enabled for boot.
4. More than 1 ms later, the PMIC enables power to 2.8V/3.3V I/O rails that need to be enabled for boot.
5. The PMIC drives a 32.768 kHz clock to Tegra 4 devices.
6. More than 1 ms later, the PMIC deasserts reset to Tegra 4 devices. The 32.768 kHz clock and all power rails must be stable and at their nominal value before reset is deasserted.
7. The PMC block resets and deasserts reset to the CAR block, which resets the rest of the Tegra 4 chip.
8. Tegra 4 devices begin boot from internal ROM.

Figure 26: Frozen Boot Sequence



10.4.2 Deep Sleep Wake-up Sequence

During Deep Sleep, VDD_CORE and VDD_CPU power rails are powered off. PMC logic resident in the always-on RTC power domain remains powered, detecting wake signals to initiate bringing the Tegra 4 device out of the Deep Sleep state. I/O rails associated with wake-up signals need to remain powered during Deep Sleep.

A full Deep Sleep / Wake-Up sequence proceeds according to the following steps:

Setting up Deep Sleep

1. Software configures the PMC CPU and core power request signal polarities and enables.
2. Software disables unneeded logic blocks and peripherals and disables their clocks.
3. Software programs the PMC with the wake-up condition.
4. Software programs scratch registers with information needed to restore state after wake-up.
5. Software interacts with the PMC to establish the idle conditions on the I/O.
6. Software programs the external SDRAM to enter self-refresh mode.
7. Software programs the PMC DPD to sample the state of all I/O lines. The sampled line states will remain driven during Deep Sleep.

8. Deep Sleep is typically entered when the last processor goes idle and managed by the Tegra 4 flow controller. Software needs to program the PMC to enter Deep Sleep prior to the last processor executing the WFI/WFE instruction used to signal the flow controller to initiate the transition
9. The PMC asserts the clamping signals for all power-gated islands.
10. The PMC deasserts CPU and core power requests.
11. The PMIC removes VDD_CORE and VDD_CPU power rails, and Deep Sleep is entered.

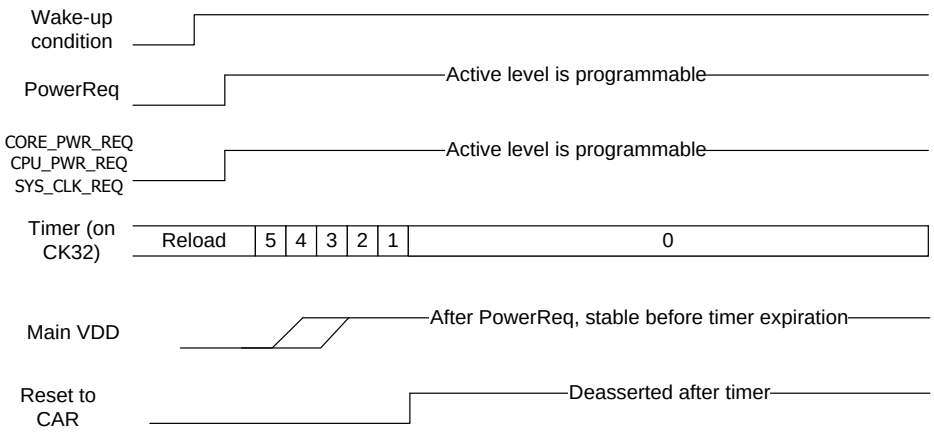
Note: To power-off VDD_CPU:
- Assert resets to CPU via CLK_RST_CONTROLLER_RST_CPU_CMPLX_SET
- Disable clock to CPU via CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0_CLK_ENB_CPU
- Remove VDD_CPU

Deep Sleep Wakeup

1. The PMC detects a wake-up condition and latches the condition into a register.
2. The PMC asserts the core power request signal, asserts reset to the CAR, and starts a timer (power good delay).
3. The PMIC restores VDD_CORE and optionally VDD_CPU.
4. The power good timer expires, and the PMC releases the reset to main.

Note: To power-on VDD_CPU:
- Restore VDD_CPU
- Enable clock to CPU via CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0_CLK_ENB_CPU
- If using PLLX as CPU clock source is desired, configure/enable PLLX and switch to PLLX (this step can be done after step 4).
- Deassert resets to CPU via CLK_RST_CONTROLLER_RST_CPU_CMPLX_CLR

Figure 27: Deep Sleep Wake-Up Sequence



10.4.3 Suspend Mode Wake-up Sequence

In the Suspend Mode wake-up sequence, the PMC needs to wake up the CPU by removing the power-gating:

Suspend Setup:

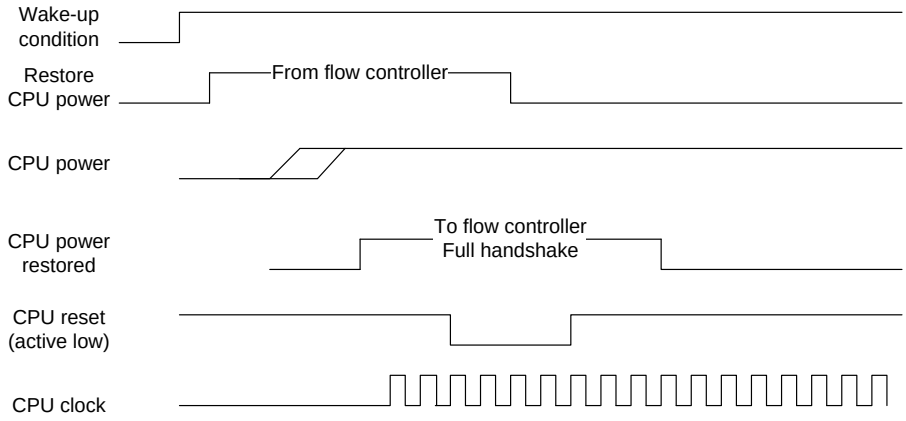
1. Software configures the Tegra flow controller with the Suspend Mode wake-up conditions.
2. The flow controller interacts with the CPU to ensure that no transactions are in flight.

3. The flow controller indicates to the CAR to reset the CPU.
4. The PMC clamps the CPU signals and removes power from the CPU.

Suspend Wakeup

1. The flow controller detects a Suspend mode wake-up condition.
2. The flow controller sends a trigger event to the PMC.
3. The PMC restores power to the CPU.
4. The PMC sends an acknowledge to the flow controller.
5. The flow controller restores the clock to the CPU.
6. The CAR asserts reset to the CPU.
7. The PMC removes clamping to the CPU.
8. The CAR deasserts reset to the CPU.

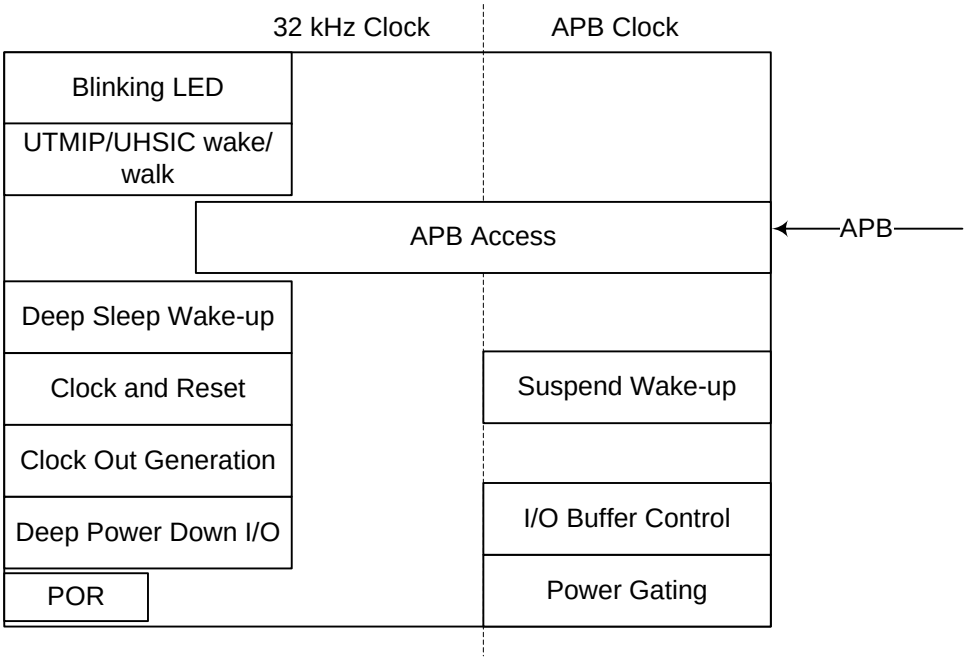
Figure 28: Suspend Mode Wake-Up Sequence



10.5 Reference Block Decomposition

The next figure shows a reference decomposition of the PMC.

Figure 29: Reference Block Decomposition Diagram



The blocks inside the PMC are:

- Wake-up logic for Deep Sleep and Suspend modes
- Power-gating logic
- External clock and reset
- Blinking LED
- I/O buffer control
- APB interface
- UTMIP/UHSIC wake/walk logic
- Deep power down I/O logic (independent on LP0)
- Clock out generation

10.5.1 Deep Sleep Entry Logic

There are two ways to enter Deep Sleep mode:

- Writing into the DPD register
- Interaction with the power gating logic when the DPD trigger enable (side effect) bit is set.

The second method is the recommended usage as described before. The exact sequence was described in more detail earlier in this section.

Note that entering Deep Sleep mode should happen after an ordered shutdown of Tegra 4 devices. As part of the shutdown procedure, external interfaces must be placed in their idle condition. The PMC provides logic that ensures these interfaces maintain their idle state during Deep Sleep itself. Software is responsible for sampling the idle state by writing to the DPD sample bit before entering Deep Sleep mode.

Software ensures that I/O paths that must remain active during Deep Sleep mode are identified as such, by correct configuration of ownership bits in a PMC register. These ownership bits are present for the signals used by the KBC (where the size of the scan matrix is defined by the customer, the exact required size is reflected in the ownership bits) and for the clock request signal (that may not be required depending on the customer clock logic).

Software must also ensure that wake-up conditions are programmed properly. Wake-up level and wake-up mask should reflect expected wake-up events. Power Good Timer and Power Down Timer must be set according to the PMIC Specification.

The Deep Sleep mode entry/wake-up logic operates largely on the 32.768 kHz clock, so software must respect a minimal delay between a Deep Sleep mode exit and before issuing the next Deep Sleep mode entry command. This delay is of the order of 2 periods of the 32.768 kHz clock. This should never be a problem given that the code to restore the processor state requires much more time.

There are two ways to enter Deep Sleep mode:

- Option 1 (side effect): First CPU power is gated, and as a result (“side effect”), the PMC performs an orderly LP0 entry. The PMC hardware also sets PMC_DPD_ENABLE (automatically). Normally, this is the only option for LP0 entry.
- Option 2 (direct write): Writing directly into the PMC_DPD_ENABLE register. This is not a safe option and is only for debug or diagnostic workaround purposes.

10.5.2 Deep Sleep Wake-Up Logic

The wake-up logic performs the following tasks:

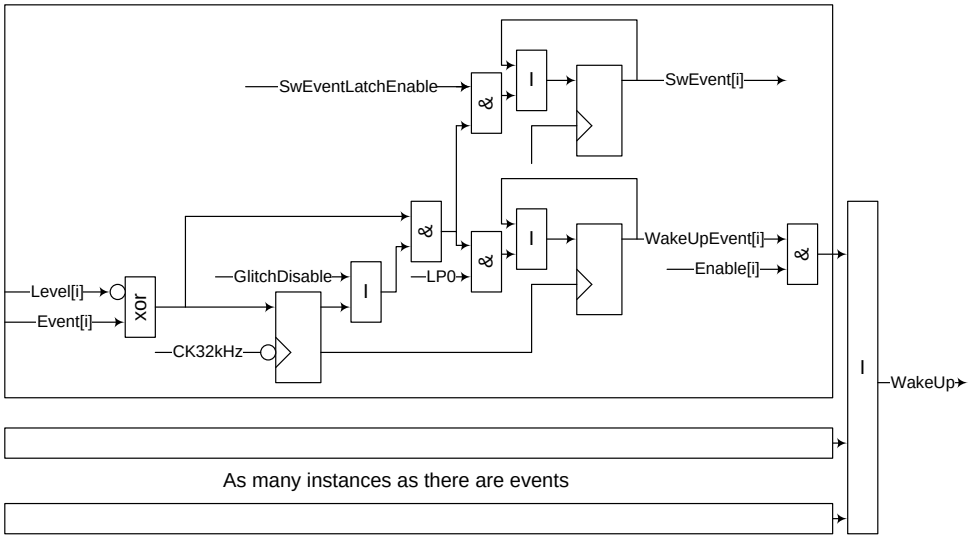
- Tracks the state of a programmable set of wake-up conditions
- Detects a wake-up condition
- Starts the sequence of action required by the Deep Sleep mode wake-up

A wake-up event is either a change of level on one in a set of specified pins or an assertion of RTC or KBC interrupts.

The logic that tracks for change of levels in external pins contains a glitch filter that may be optionally disabled. Internal interrupts do not require deglitching logic, but are passed through the same logic, as this is also the synchronization logic. A conceptual schematic is shown in the next figure. The different stages are:

- Normalizing the active level to high by inverting the wake-up signals identified as active low
- Latching either the direct signal or a deglitched version of it (global enable)
- Masking the latched signals to generate the wake-up signal

Figure 30: Deep Sleep Wake-up Logical Circuit



Note that the WakeUp signal shown is not directly the CORE_PWR_REQ signal, and there are two further processing steps defining the CORE_PWR_REQ signal:

- The polarity of the CORE_PWR_REQ signal is programmable
- CORE_PWR_REQ signal is disabled at reset. It should be enabled only after polarity is set accordingly to PMIC specification

Furthermore, there is a System Clock enable signal that is equal to CORE_PWR_REQ, but with its own independent control bits for polarity and output enable.

The set of latched wake-up events are readable by software once the processors are active. The logic accumulates all events that take place while in Deep Sleep mode. Normally only one enabled event is present in the set, but simultaneous events could take place. Each bit of the latched wake-up event register is cleared by writing a 1b to it.

A second set of latches is also present, with the same structure as the wake-up event register but under software control. The most important difference between the two sets is that events that occur outside of the Deep Sleep mode will be captured in the second set, including:

- Events that could occur during the Deep Sleep mode transition itself, i.e. before the hardware enters the Deep Sleep state, but after software has instructed hardware to enter Deep Sleep.
- Events that occur after the hardware wake-up, but before software has enabled the peripheral functions in the NPG that process the signals tied to the wake-up events, i.e., the GPIO event logic.

10.5.2.1 Deep Sleep Wake-up Event Input Path

The wake-up signals can be configured to be level (active high or low based on PMC_WAKE*_LVL registers) or pulse (based on WAKE2_LVL[ALLOW_PULSE_WAKE]). However, there has to be a minimum delay between two consecutive wake events handling (which is programmed in PMC_WAKE_DELAY). A level wake-up signal must be valid when the PMC can act upon it. For pulse wake-up signals, the PMC registers them even if they trigger during PMC_WAKE_DELAY window. However, the PMC will act upon a (registered) pulse event only after the PMC_WAKE_DELAY window has expired.

The PMC detects wake signals by using 32 kHz logic. Therefore, in either case (level or pulse), a signal has to last long enough for the PMC to recognize the change in signal. In addition, the PMC provides an option to filter out glitches.

The wake-up logic as defined, recognizes wake-up signals matching the programmed wake-up active level that exceed certain time limits and rejects events less than another limit as defined in the following table.

Table 29: Deep Sleep Wake-up Logic

Glitch disable bit	Never wake-up if event is less than	Always wake-up if more than
0 (Glitch reject)	0.5 period of 32.768 kHz clock = 15.25 µs	1.5 period of 32.768 kHz clock = 45.755 µs
1 (No glitch reject)	0 period of 32.768 kHz clock = 0 µs	1 period of 32.768 kHz clock = 30.50 µs

Between these two values, the probability of detection increases linearly from 0 to 1 when events are asynchronous to the 32.768 kHz clock. The latency delay is also a probabilistic function, varying linearly between the two limits presented in the above table.

10.5.3 Suspend Wake-up Logic

The only Suspend mode wake-up specific logic in the PMC is related to the power gating logic as described previously.

10.5.4 CPU Rail Partitions

CPU power-gating control is separated from SoC power-gating control because the CPU rail partitions need different inter-zone delay values (configured in PMC_PWRGATE_TIMER_* registers). There are 8 zones per CPU partition.

The following table provides a list of CPU rail partitions.

Table 30: CPU Rail Partitions

Partition	Partition Description	Notes
CE0	Cluster0 CPU0	Normally, power-gated on/off via flow-controller interaction. Can be power-gated on/off by direct register write. Uses PMC_PWRGATE_TIMER_CE* inter-zone delays.
CE1	Cluster0 CPU1	
CE2	Cluster0 CPU2	
CE3	Cluster0 CPU3	
C0NC	Cluster0 Non-CPU	
CRAIL	CPU Rail	Normally, power-gated on/off via flow-controller interaction. Can be power-gated on/off by direct register write. On/Off delays are programmed in PMC_CPUPWRGOOD_TIMER and PMC_CPUPWROFF_TIMER registers

10.5.5 SoC Rail Partitions

In SoC partitions, only CPULP and C1NC can be power-gated on/off via the flow controller. All other partitions can only be power-gated (on/off) by PMC register writes. Tegra 4 PG partitions have 8 zones per partition.

Partition	Partition Description	Notes
CELP	Cluster1 CPU	Normally, power-gated on/off via flow-controller interaction. Can be power-gated by direct register writes. Uses PMC_PWRGATE_TIMER_* inter-zone delays.
C1NC	Cluster1 Non-CPU	
TD	3D Engine	Power-gated only by direct register writes. Uses PMC_PWRGATE_TIMER_* inter-zone delays. In separate register is used for inter-partition delays GPU partitions are sequenced (one after another of this partition on/off request. There are a total of 6 partitions in the GPU.

Partition	Partition Description	Notes
n/a	3D Engine2	Power-gated only by direct register writes. Uses PMC_PWRGATE_TIMER_* inter-zone delays.
VE	CSI,ISP,VI	
VDE	Video Decode	
MPE	Video Encode	
HEG	VIC, Host1x, …	
n/a	SATA	
DIS	DisplayA	
DISB	DisplayB	
XUSBA	USB3 (XUSBA)	
XUSBB	USB3 (XUSBB)	
XUSBB	USB3 (XUSBC)	

10.5.6 PG Partitions

This subsection provides the default power, clamp, and reset status of the PG partitions.

10.5.6.1 Cold Boot (Power-on Reset)

By default, all of the CCPLEX related partitions (except the C0NC) are power-gated and non-CCPLEX partitions are power-ungated. The clamps are accordingly enabled (for the power-gated domain) or disabled (for the power-ungated domain). The C0NC is power-ungated by default (to keep the CAR reset defaults consistent with the PMC power status at LP0 exit).

Refer to the PMC_PWRGATE_STATUS and PMC_CLAMP_STATUS register descriptions for the default power and clamp status, respectively, of PG partitions.

10.5.6.2 LP0 Exit

At LP0 exit, the PMC retains power status of all the PG partitions. The PMC does not restore the CPU rail – if necessary, software needs to power-up the CPU rail during LP0 exit (via a PMC register write).

The CAR unit is powered off during LP0, so the CAR provided reset defaults at LP0 exit are the same as its cold boot defaults. At LP0 exit, the C0NC is expected to be power-ungated. Therefore, its cold boot default is power-ungated, so its reset (which is provided by the CAR block) can be kept the same for cold boot and LP0 exit.

10.5.7 Power-Gating Logic

The power-gating logic is a simple programmable sequencer. When the power-gating logic is triggered, it sequences the set of power-gating signals with a programmable period (measured in cycles) between the toggling of consecutive power-gating signals.

There are three power-gated domains. For all of them, the trigger can be a register write (see register definitions). The CPU power-gated domain has an extra trigger coming from the flow controller. The PMC and flow controller provide a full handshake, with the PMC providing an acknowledge back to the flow controller when the CPU power has completed its transition (this would normally only be for OFF to ON transition at Suspend mode wake-up).

The power-gating logic operates on the APB clock. The timing sequence may be dependent on the exact frequency, with a hardware limit that the minimum transition period is at least 64 ns.

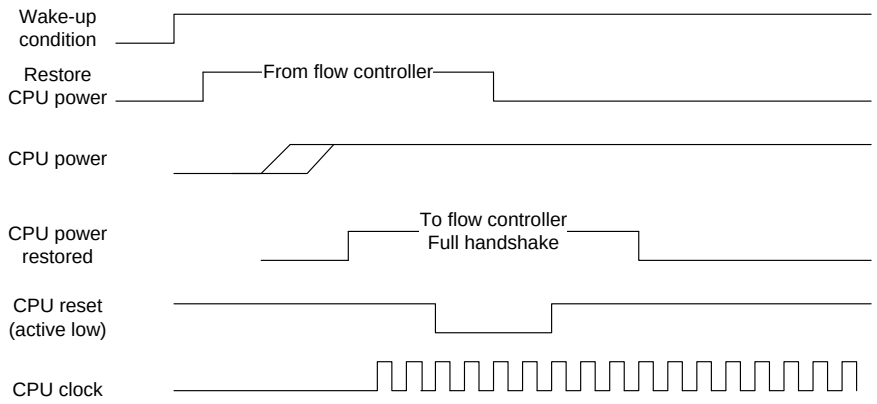
The power-gating logic may be triggered either by direct register writes using the PMC toggle register or by interaction with the flow controller. The second control path is used when entering Deep Sleep or Suspend mode to allow for an orderly shutdown of the processor. The sequence can be summarized in the following manner:

1. The CPU decides to enter either Deep Sleep or Suspend mode.
2. In both cases, power to the CPU must be gated off.
3. The CPU does any required clean-up. This is especially important for Deep Sleep (all interfaces in idle, save context, etc.).
4. The CPU informs the flow controller it wants to remove its power with two variants:
   - For Suspend mode, the CPU also defines the set of flow controller events that will wake up the CPU (the flow controller is not power gated together with the CPU).
   - For Deep Sleep, a write to a DPD sample register preserves the I/O state to be driven when the chip enters Deep Sleep mode.
     For Deep Sleep, the CPU must define in the PMC the set of wake-up events for the whole Tegra 4 device and set the DPD trigger enable bit.
5. The CPU enters an endless loop, executing the WFI/WFE instruction.
6. The flow controller asserts reset to the CPU.
7. The flow controller informs the PMC that CPU power-gating OFF is requested.
8. The PMC power gates the CPU OFF, then informs the flow controller.
9. If the DPD trigger enable was set, the PMC also performs a Deep Sleep entry, which will result after a while in the core power disappearing.

The Deep Sleep wake-up was described before and is essentially a full reset of the core logic. The Suspend mode wake-up logic includes an interlock with the flow controller that works like this:

1. The flow controller detects the Suspend mode wake-up condition.
2. The flow controller informs the PMC that power-gating ON is requested.
3. The PMC power gates the CPU ON, then informs the flow controller.
4. The flow controller deasserts the reset to the CPU.
5. The CPU checks different status registers to establish this was in fact a Suspend mode wake-up.

Figure 31: Power-Gating Logic



10.5.7.1 Software and Hardware PG Request Interlocking

The PMC power-gating controller can only power gate (or ungate) one partition at a time. However, the PMC can get independent PG requests from software by a direct register write and hardware through the flow controller.

In Tegra 3 devices, the PMC drops the PG request when it is busy sequencing a hardware-initiated request. In Tegra 4 devices, the behavior of the PMC_PWRGATE_TOGGLE[START] register bit is changed as follows:

- Software sets this bit to '1' to request a power-gate toggle.
- Hardware clears this bit to '1' when hardware starts to execute a power-gate toggle request.

For backward-compatibility, the PMC_PWRGATE_TOGGLE[START] register bit can be written to '0' by software. Normally, software should never write this bit to '0'. If software writes it to '1' and then writes it to '0' before hardware starts to execute the request, the software request gets dropped:

10.5.7.2 GPU Power Gating

Internally, the Tegra 4 3D engine has 6 partitions which are seen as one power partition by software (TD). Due to power restrictions, the 6 partitions should be brought up/down sequentially. This operation is normally transparent for software.

For debug purposes, the PMC provides support for sequencing all 6 partitions at the same time. The TD_PWRGATE_INTER_PART_TIMER register programs the period between each consecutive partition's power-gate up/down sequence. Partitions are powered up in order: TDA, TDB, TDC, TDD, TDE, and TDF. Power-gating down is executed in reverse order. Programming a 0 value in the TD_PWRGATE_INTER_PART_TIMER register will trigger simultaneous power-gating of all 6 partitions.

10.5.8 I/O Buffer Control

Deep Power Down I/O buffer management ensures that the I/Os are maintained in an idle state while in Deep Sleep mode. The procedure has been described before and requires software sequencing:

1. Software makes sure that the affected I/O ports are in their idle state, i.e., the state that will persist during Deep Sleep.
2. When software sets the DPD sample register , the idle state of the I/O control signals is captured, i.e., the driving direction and the data driven if the direction indicates output.
3. Software writes the DPD enable bit and muxes inside the I/O buffers now drive the stored idle state, while the corresponding control signals coming from the core are ignored.
4. Software triggers the Deep Sleep mode, killing itself.
5. Hardware detects the Deep Sleep wake-up condition.
6. Software wakes up and restores the state.
7. Software clears the DPD sample bit.
8. I/O control can be sequenced back to the controlling block once software has initialized the core block to drive outputs to the same idle state they maintained before entering DPD.

The PMC contains a software programmable register, with one bit per I/O rail. Each bit disables internal logic inside the I/O buffer that would otherwise consume power when the I/O power is OFF. Each bit should be set by software before shutting off the corresponding I/O power. Conversely, each bit should be reset after transitioning the I/O voltage to ON (and before enabling the set of interfaces using the corresponding I/O rail).

The PMC contains two registers related to 3.3V I/O support. Specific power detector cells per I/O voltage domain indicate if their supply voltage is low or high. The detector cells consume significant power, so they should not be left on all the time. The PMC contains a register enabling the detector cells.

10.5.9 Power Down I/O Logic

Tegra 4 PMC provides support for placing interfaces into a deep power down mode outside the Deep Sleep state. DPD domain groups include:

CSIA, CSIB, DSI, POP_CLK, POP_ADDR_CMD, DDR_ADDR_CMD, DISC_ADDR_CMD, DDR_DATA, SYS, NAND, UART, BB, VI, AUDIO, LCD, SD, IPI_BIAS, PEX_BIAS, PEX_CLK1, PEX_CLK2, COMP, POP_VTTGEN, DISC_VTTGEN, DAC, USB0, USB1, USB2, USB_BIAS, HSIC, HDMI, SDMMC1, SDMMC3, SDMMC4, CAM, PEX_CNTRL

- **DISC_VTTGEN** includes mem_addr_vttgen_1_pad, mem_addr_vttgen_2_pad, mem_vttgen_26_pad, mem_vttgen_27_pad, mem_vttgen_28_pad, mem_vttgen_29_pad
- **COMP** includes xm2_comp_pd_pad, xm2_comp_pu_pad
- **POP_CLK** includes xm2_mclk_addr_pad
- **POP_ADDR_CMD** includes dvi_d10_pad, dvi_d11_pad, dvi_vsync_pad, dvi_hsync_pad dvi_d0_pad, dvi_d1_pad, dvi_d2_pad, dvi_d3_pad, dvi_d4_pad, dvi_d5_pad, dvi_d6_pad, dvi_d7_pad, dvi_d8_pad, dvi_d9_pad
  DDR_ADDR_CMD includes xm2_mcas_n_pad, xm2_mras_n_pad, xm2_mwe_n_pad, xm2_mba[2:0]_pad, xm2_ma[14:10]_pad,xm2_odt0_pad, xm2_odt1_pad, xm2_reset_pad
- **DISC_ADDR_CMD** includes xm2_mclk_pad, xm2_mcke0_pad, xm2_mcke1_pad, xm2_mcsc_n_pad, xm2_mcsd_n_pad, xm2_ma[9:0]
- **DDR_DATA** includes xm2_data0_pad, xm2_data1_pad, xm2_data2_pad, xm2_data3_pad, xm2_mckfb0_pad, xm2_mckfb1_pad, xm2_mckfb2_pad, xm2_mckfb3_pad

Entering DPD mode for a single DPD domain has simplified steps of LP0 entry. To set a domain in DPD mode:

- Program the interface to drive the idle state out
- Set the sample register (PMC_DPD_SAMPLE)
- Set the fields corresponding to the DPD domains to 1 in the PMC_IO_DPD_REQ register or the PMC_IO_DPD2_REQ register and CODE to DPD_ON
  - Fields set to 0 indicate no status change
  - IO_DPD_STATUS and IO_DPD2_STATUS are read-only registers that display the status of each rail. The PMC_DPD_SAMPLE register must be reset before the next sample operation.
- Set PMC_SEL_DPD_TIM to the value corresponding to minimum 80 ns (this allows for proper timing sequence between sel_dpd, e_dpd)
- Set the interface to drive the idle state
- Set the fields corresponding to the DPD domains to 1 in the PMC_IO_DPD_REQ register or the PMC_IO_DPD2_REQ register and CODE to DPD_OFF
- Set the tristate field in the miscellaneous register to disable for all pads in the set of affected domains (this will deassert sel_dpd in the mini pad macros). Some bricks do not have mini_pad_macros. For those, this step will be skipped.
  - Fields set to 0 indicate no status change

The following DPD domains do not have mini pad macros or do not drive the state out during deep power down mode:

MIPI_BIAS, PEX_BIAS, PEX_CLK1, PEX_CLK2, COMP, POP_VTTGEN, DISC_VTTGEN, DAC, USB0, USB1, USB2, USB_BIAS, HSIC, HDMI, DDR_ADDR_CMD, DISC_ADDR_CMD, DDR_DATA, POP_CLK.

For the above domains the sequence of entering and leaving DPD mode is simpler:

To set the domain in DPD mode:

- Set the fields corresponding to the DPD domains to 1 in the MC_IO_DPD_REQ register or the MC_IO_DPD2_REQ register and CODE to DPD_ON
  - Fields set to 0 indicate no status change

To reset DPD mode for the DPD domain:

- Set PMC_SEL_DPD_TIM to the value corresponding to minimum 80 ns (this allows for proper timing sequence between sel_dpd, e_dpd)
- Set the fields corresponding to the DPD domains to 1 in the PMC_IO_DPD_REQ register or the PMC_IO_DPD2_REQ register and CODE to DPD_OFF

- Fields set to 0 indicate no status change

Note: In the current architecture only one sample signal exists to perform sampling while entering dpd_mode. This means that all interfaces are sampled at the same time. A sequence of multiple DPD sets requires keeping all interfaces already in DPD mode, and driven as idle. *We can either add a sample bit per interface, or try to modify mini pad macros to prevent sample for pads which are already in DPD mode*.

For regular LP0 mode, all pads enter/exit DPD at the same time.

Note about VI (called PVI): all pads in this group with exception of sdmmc2_vi_sclk_lb, dvi_mclk, dvi_pclk are covered by 2 groups –POP_ADDR_CMD and POP_CLK. So to fully turn off VI, all three groups need to be turned off.

The new registers associated with this feature:

- PMC_IO_DPD_REQ
- PMC_IO_DPD_STATUS
- PMC_IO_DPD2_REQ
- PMC_IO_DPD2_STATUS
- PMC_SEL_DPD_TIM

10.5.10 Clock Control Logic

Tegra 4 devices support enabling an external oscillator to run in Deep Sleep (LP0) mode. It is combined with the ability to output clock (CLK_OUT) which can be extended outside LP0. See the PMC_OSC_EDPD_OVER and PMC_CLK_OUT_CNTRL registers for register descriptions.

There are two functions embedded in this feature:

- Programmable option of keeping the oscillator ON during DPD mode
- For low-cost systems, the ability to output the clock on dap_mclk1, dap_mclk2, clk3_out pins.

Programmable option of keeping oscillator ON during DPD mode

The OSC_CTRL_SELECT bit in the PMC_OSC_EDPD_OVER register configures whether or not the PMC starts controlling the external oscillator. Setting the EN bit to 0 (disabling the oscillator) should only be used during deep power down mode with automatic enabling on leaving LP0 (this is to avoid system lockup). Based on PMC_OSC_EDPD_OVER setup, the external oscillator can be controlled as follows:

- Completely controlled by the CAR, entering deep power down mode on LP0 (shutting down oscillator)
- Controlled by the PMC, entering deep power down mode on LP0
- Controlled by the PMC, not entering deep power down mode, but the oscillator output is disabled (used to accelerate warm boot time)
- Controlled by the PMC, not entering deep power down mode, the oscillator output enabled – used for clock out feature.

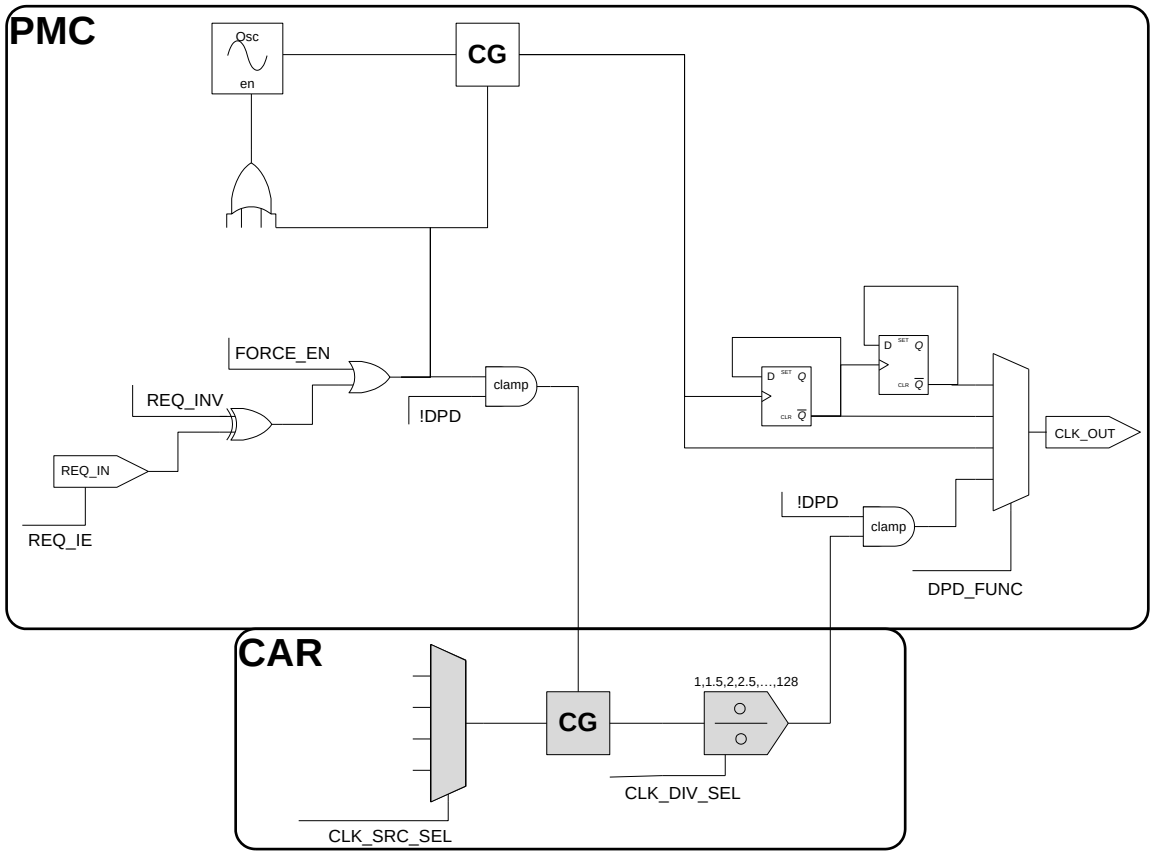
For low-cost systems, ability to output clock on dap_mclk1, dap_mclk2, clk3_out pins

The Clock out function is illustrated below. Three independent request lines (dap_mclk1_req, dap_mclk2_req, clk3_req) are used to assert clock requests on the corresponding clock outputs: dap_mclk1, dap_mclk2, clk3_out. When such a request is decoded, the PMC switches controls on new clock pad_macros to bypass regular pinmux route and output clock instead. There are a variety of options available for decoding the clock request and for the source of the clock output itself.

The request can be decoded as active high, low, or the clock output could be forced out independently of the state of request line. The clock output line, when idle, could be driving out 0, 1, or be tristated. Once the request is accepted, the clock source

could be a CAR output or an oscillator output divided by 1, 2, or 4. The request can be issued outside or in LP0, and the only difference in behavior should be that during LP0 CAR output as clock source is not available.

Figure 32: CLK_OUT Paths



10.5.11 Resets

The PMC receives the primary chip reset (from the SYS_RESET_N_ pad) and generates various resets for itself, the KBC, the RTC, the CAR, etc. From the PMC provided reset, the CAR unit generates resets for most of the units in the chip.

In addition to chip reset, the PMC receives other events (thermal, WDT, SW, LP0 wake), which also result in variants of the system reset.

10.5.12 Clamping

When a partition is power gated OFF, all signals leaving that partition are clamped to their idle value.

Applying clamping presents no problems, because all blocks into the partition are powered OFF and assumed to be in reset prior to the power gate transition.

Removing the clamping is more complex because Tegra 4 devices use a synchronous reset strategy, so the correct sequence is the following:

- Restore power, reset and clamping are both active at that time
- Restore clocks, so that reset conditions propagate
- Remove clamping
- Remove reset

For maximum flexibility, removing the clamping can be controlled by software. Writing to a trigger register removes the clamping for the identified set of partitions. This does not work for Suspend mode exit, where the CPU must be restarted by hardware only.

In Suspend mode exit, there is an interlock between the Clock And Reset (CAR) block and the PMC, so that the correct sequence is performed. The CAR indicates to the PMC when the CPU clamping may be removed. This signal is asserted after the clock to CPU is restarted. The reset to CPU will only be removed after a known delay following the request to remove the clamping. The PMC removes the CPU clamping as soon as it receives the request to do so. This is similar to the interlock between the PMC and flow controller, except that this is not a full handshake.

10.5.13 Scratch Registers

The PMC provides a number of scratch, secure scratch, and bondout registers. The scratch registers are used to save some of the system context information during LP0.

There are 120 PMC scratch registers and 24 secure registers. To save area, secure scratch registers 4-23 and scratch registers 1-119 are implemented in RAM. Due to the requirement of being backward compatible with prior Tegra device's address space, scratch registers do not have continuous address space.

Secure scratch registers are general-purpose read/write registers that can be locked by secure/trusted code. The read/write lock bit is reset by cold and LP0 resets.

10.5.14 Blinking

Tegra 4 devices support a blinking LED whenever the cell phone is not powered OFF, including in Deep Sleep mode. The blinking LED is controlled from the RTC partition.

The PMC contains a counter with two programmable values indicating the On and Off periods (in multiple of 16 cycles of the 32.768 kHz clock). The counter is 16 bits long, for a maximum On or Off period of 2^{16} * 16 / 32.768 10^{3} or about 16 seconds.

10.5.15 Pinmuxing

Some of the signals controlled by the PMC may not be required in certain customer designs, especially:

- The system clock may not support an enable function, so the system clock enable (SYS_CLK_REQ) is not used.
- The system may not require a LED active in Deep Sleep or another device (BB) may drive the LED, so the power LED is not used.

It is important to allow the corresponding balls to be reused for other purposes via pinmuxing. This is not directly a responsibility of the PMC but the correct pinmuxing must be present. The PMC provides a set of ownership registers that control if the RTC logic (PMC or KBC) has control of pins with multiple uses (KBC, PMC, or typically GPIO), some of which are not controlled by the RTC logic. The ownership bit controls a final stage of muxing on top of the more general pinmuxing stage.

As an important special case, the system clock enable is one of the Deep Sleep wake-up event pins. Obviously usage as a wake-up event pin is incompatible with usage as a system clock enable pin.

10.5.16 APB Access

The APB access logic is complicated by the need to not unduly block the bus for extended periods of time even when accessing registers that are logically in the 32 kHz domain. The following principles are used to avoid any delay when accessing registers logically in the 32 kHz domain:

- Write operations always take place in the APB clock domain; i.e., the corresponding registers are placed in the APB clock domain and passed as wires to the 32 kHz domain where they are used without further synchronization. This only works because values passed in this manner are considered essentially static. There is an insignificant

probability that the 32 kHz logic sees an intermediate value of a multiple bit field, but an enumeration of the affected fields shows that this cannot result in any significant problem at the system level.

- Read operations are to shadow registers in the APB clock domain. The shadow registers continuously reflect the values of the equivalent registers found in the 32 kHz clock domain. Only registers reflecting status present in the 32 kHz domain are affected.

10.6 Sensor Reset

The PMC monitors the temperature sensor signal. When the signal becomes active, the PMC resets the whole chip and might also power gate CPU1-3. Both reset and power gating are configurable and can be enabled or disabled. Reset of the whole chip automatically causes CPU1-3 power gating and request for turning down cpu_power.

Power gating is a simplified version of the regular power-gating function. In the first cycle, clamping is enabled. In the next cycle, CPU1-3 are power gated. The above approach simplifies the logic and allows for performing the function before apb_clk is shut down by reset.

The reset status register bit is set to indicate that the thermal sensor event has happened. If the PMC is entering LP0 mode when the temperature sensor triggers, the reset as a direct result of the active sensor will not happen (to avoid destruction of the PMC state). However, the process of entering LP0 causes a reset a few cycles later. CPU1-3 will be shut down in each situation.

10.7 Register Definitions for the PMC

To speed up operation, the PMC register file operates in the local peripheral interface bus domain (APB) rather than in the 32 kHz clock domain used for PMC processing.

Registers in the PMC control entering/exiting Deep Sleep/Suspend mode, power detect, and I/O power functions. They also control CORE_PWR_REQ, SYS_CLK_REQ, and LED_BLINK pins.

The following registers should be programmed/read for different power events in order for PMC to function properly.

IMPORTANT:	**NO_IO_POWER REGISTER (APBDEV_PMC_NO_IOPOWER_0)** This register is used to disable pads that have I/O power turned off. When an I/O power rail is turned off, ramping up, or no longer used and ready to turn off, the corresponding bit in this register should be set to 1. The bit only needs to be turned on when the I/O power rail is stable and you are ready to enable some interface on the I/O power rail. Leaving no_io_power set to 0 **for greater than 500 ms** when the rail is off may reduce the lifetime of the pad. Setting it to 1 when the power rail is on will disable pad from voltage stress.

BEFORE/ON ENTERING DEEP SLEEP MODE

- PMC Control Register (bits PWRREQ_POLARITY, PWRREQ_OE, SYSCLK_POLARITY,SYSCLK_OE, LATCHWAKE_EN)
- WAKE_MASK
- WAKE_LVL
- DPD_PADS_ORIDE (for required overrides)
- PWRGOOD_TIMER
- DPD_SAMPLE
- DPD_ENABLE

AFTER WAKE-UP FROM DEEP SLEEP

- WAKE_STATUS
- SW_WAKE_STATUS
- DPD_ENABLE
- PMC Control Register (bit LATCHWAKE_EN)

BEFORE/ON POWERING DOWN PARTITIONS

- PWRGATE_STATUS
- PWRGATE_TIMER_OFF (for power down)
- PWRGATE_TIMER_ON (for power up). This has been deprecated.
- PWRGATE_TOGGLE
- REMOVE_CLAMPING_CMD (for power up)

FOR POWER DETECT SEQUENCE

- PWR_DET
- PWR_DET_LATCH

FOR BLINK

- BLINK_TIMER
- PMC Control Register (bit BLINK_EN)
- DPD_PADS_ORIDE (for blink field)

All other operations require single register access.

10.8 Register Differences for Tegra 4 Devices

The following list indicates register differences in Tegra 4 devices compared with prior Tegra devices:

- PMC_SWRST function has been removed
- MAIN_RST bit in the PMC control register resets whole chip
- SEC_DISABLE registers now have granularity added per register
- WAKEUP events assignment changed
- PWRGATE_TOGGLE has more partition IDs
- REMOVE_CLAMPING_CMD has more fields due to more power partitions change

- POWER_GATE_STATUS has more fields due to more power partitions change
- NO_IOPWR has more fields due to new power rails
- PWR_DET has more fields due to new power rails
- Increased number of Bondout registers
- Increased number of Scratch registers
- PG_MASK new fields added for new power-gated domains
- PLLP_WB0_OVERRIDE new fields for M and U PLLs

10.9 PMC Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

10.9.1 APBDEV_PMC_CNTRL_0

This register controls the clock to the KBC and RTC, and the reset to the CAR, KBC, and RTC. It also manages polarity and output enable of the sys_clk_req and core_pwr_req pins. At reset, both are disabled since the PMIC properties are unknown.

Auxiliary functions include enabling of blinking functions, side-effect option, and software wake-up latching.

The LP0 entry is to be triggered by the PMC side-effect option. Software needs to configure the PMC for the side-effect option before triggering the power off of last CPU. If the side-effect option is set, then the PMC would initiate LP0 in the following cases:

- A power-off request for the CPU rail will lead to CPU rail power-off followed by LP0 entry.
- A power-gating request for cluster1 non-CPU (i.e., C1NC) will lead to power-gating of the C1NC followed by LP0 entry

PMC Control Register

Offset: 0x0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
21:20	0x0	CPUPWRGOOD_SEL: CPUPWRGOOD pin select. There are 4 potential source pins (soc_therm_oc{1,2,3,4}) used as PWRGOOD. This field needs to be programmed (at boot) based on which pin is used as PWRGOOD. 0 = SOC_THERM_OC1 (default) 1 = SOC_THERM_OC2 2 = SOC_THERM_OC3 3 = SOC_THERM_OC4
19	0x0	CPUPWRGOOD_EN:CPU_PWR_GOOD (From PMIC to PMC) signal is enabled. 0 = DISABLE 1 = ENABLE
18	0x0	FUSE_OVERRIDE: Fuse override 0 = DISABLE 1 = ENABLE
17	0x0	INTR_POLARITY: Inverts INTR polarity 0 = DISABLE 1 = ENABLE
16	0x0	CPUPWRREQ_OE: Power request output enable. Resets to tri-state 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
15	0x0	CPUPWRREQ_POLARITY: Inverts power request polarity 0 = NORMAL 1 = INVERT
14	0x0	SIDE_EFFECT_LP0: When set, causes the side effect of entering LP0 after powering down the CPU 0 = DISABLE 1 = ENABLE
13	0x0	AOINIT: AO initialized purely software diagnostic and interpretation 0 = NOTDONE 1 = DONE
12	0x0	PWRGATE_DIS: Disable power gating - global override, will override function of PWRGATE_TOGGLE register. All partitions will stay enabled. 0 = DISABLE 1 = ENABLE
11	0x0	SYSCLK_OE: Enables output of system enable clock - works only if the SYS_CLK field in DPD_PADS_ORIDE is set to 1. 0 = DISABLE 1 = ENABLE
10	0x0	SYSCLK_POLARITY: Inverts SYS_CLK_REQ enable polarity 0 = NORMAL 1 = INVERT
9	0x0	PWRREQ_OE: CORE_PWR_REQ output enable. Resets to tristate 0 = DISABLE 1 = ENABLE
8	0x0	PWRREQ_POLARITY: Inverts CORE_PWR_REQ polarity 0 = NORMAL 1 = INVERT
7	0x0	BLINK_EN: Enables blinking counter and blink output works only if the BLINK field in DPD_PADS_ORIDE is set to 1. 0 = DISABLE 1 = ENABLE
6	0x0	GLITCHDET_DIS: Disable detecting glitches on wake-up event. In default operation, glitches are ignored on wakeup lines. if this bit is set to 1, glitch (event shorter than half 32 kHz clock, will be causing wake up from LP0. 0 = DISABLE 1 = ENABLE
5	0x0	LATCHWAKE_EN: Enables latching wakeup events - stops latching on transition from 1 to 0 (sequence - set to 1,set to 0) 0 = DISABLE 1 = ENABLE
4	0x0	MAIN_RST: Resets everything but scratch 0 and reset status. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x0	KBC_RST: Software reset to KBC. 0 = DISABLE 1 = ENABLE
2	0x0	RTC_RST: Software reset to RTC. 0 = DISABLE 1 = ENABLE
1	0x0	RTC_CLK_DIS: Disable 32 KHz clock to RTC. 0 = DISABLE 1 = ENABLE
0	0x0	KBC_CLK_DIS: Disable 32 kHz clock to KBC. 0 = DISABLE 1 = ENABLE

10.9.2 APBDEV_PMC_SEC_DISABLE_0

On separate reset (same as CAR), this register disables access (read/write to secure scratch registers). Once writes are disabled, secure registers cannot be written until power on reset or reset when exiting Deep Sleep mode.

Once reads are disabled, reads from scratch registers will return 0 until power on reset or reset when exiting Deep Sleep mode.

Single register can be disabled for reads/writes; bits 0 and 1 have an overwrite function.

Secure Register Disable

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
21	0x0	TSC_NS_WRITE: Non-secure (i.e., non-TZ) write-disable for the PMC_TSC_MULT register and the PMC_DPD_ENABLE[TSC_MULT_EN] register-bit. 0: Non-secure or secure can write. 1: Only secure can write. This bit is write-once only. Once it is set to '1', it remains '1' until reset by LP0 or system reset. 0 = OFF 1 = ON
20	0x0	AMAP_WRITE: Disable writes to the PMC_GLB_AMAP_CFG register 0 = OFF 1 = ON
19	0x0	READ7: Disable reads from secure register 7 0 = OFF 1 = ON
18	0x0	WRITE7: Disable writes to secure register 7 0 = OFF 1 = ON
17	0x0	READ6: Disable reads from secure register 6 0 = OFF 1 = ON
16	0x0	WRITE6: Disable writes to secure register 6 0 = OFF 1 = ON

Bit	Reset	Description
15	0x0	READ5: Disable reads from secure register 5 0 = OFF 1 = ON
14	0x0	WRITE5: Disable writes to secure register 5 0 = OFF 1 = ON
13	0x0	READ4: Disable reads from secure register 4 0 = OFF 1 = ON
12	0x0	WRITE4: Disable writes to secure register 4 0 = OFF 1 = ON
11	0x0	READ3: Disable reads from secure register 3 0 = OFF 1 = ON
10	0x0	WRITE3: Disable writes to secure register 3 0 = OFF 1 = ON
9	0x0	READ2: Disable reads from secure register 2 0 = OFF 1 = ON
8	0x0	WRITE2: Disable writes to secure register 2 0 = OFF 1 = ON
7	0x0	READ1: Disable reads from secure register 1 0 = OFF 1 = ON
6	0x0	WRITE1: Disable writes to secure register 1 0 = OFF 1 = ON
5	0x0	READ0: Disable reads from secure register 0 0 = OFF 1 = ON
4	0x0	WRITE0: Disable writes to secure register 0 0 = OFF 1 = ON
3:2	0x0	NOT_USED: legacy - redundant - not used
1	0x0	READ: Disable reads from all secure registers 0 = OFF 1 = ON
0	0x0	WRITE: Disable writes to all secure registers 0 = OFF 1 = ON

10.9.3 APBDEV_PMC_PMC_SWRST_0

Reset only by the POR cell; sets itself back to inactive after 2 clock cycles. Only used for emergency debugging purposes; not to be used in any functional mode.

Note: Defunct. Kept only for binary code compatibility only. Will not do anything

Register Write which Resets PMC Only

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	RST: Software reset to the PMC only 0 = DISABLE 1 = ENABLE

10.9.4 APBDEV_PMC_WAKE_MASK_0

The APBDEV_PMC_WAKE registers handle wake events whose number is less than or equal to 32. For wave events whose number is greater than 32, refer to the APBDEV_PMC_WAKE2 registers below.

Current pin assignments wake_mask, wake_status, sw_wake_status, and wake levels:

- WAKE_STATUS[0] = ulpi_data4 (wake event 0)
- WAKE_STATUS[1] = gp3_pv[1] (wake event 1)
- WAKE_STATUS[3] = sdmmc3_dat1 (wake event 3)
- WAKE_STATUS[4] = hdmi_int (wake event 4)
- WAKE_STATUS[6] = gp3_pu[5] (wake event 6)
- WAKE_STATUS[7] = gp3_pu[6] (wake event 7)
- WAKE_STATUS[8] = gmi_wp_n (wake event 8)
- WAKE_STATUS[9] = gp3_ps[2] (wake event 9)
- WAKE_STATUS[10] = gmi_ad21 (wake event 10)
- WAKE_STATUS[11] = spi1_cs2_n (wake event 11)
- WAKE_STATUS[12] = spi1_cs1_n (wake event 12)
- WAKE_STATUS[13] = sdmmc1_dat1 (wake event 13)
- WAKE_STATUS[15] = gmi_cs1_n (wake event 15)
- WAKE_STATUS[18] = pwr_int (wake event 18)
- WAKE_STATUS[19] = usb0_vbus (wake event 19)
- WAKE_STATUS[21] = usb0_id (wake event 21)
- WAKE_STATUS[23] = gmi_iordy (wake event 23)
- WAKE_STATUS[24] = gp3_pv[0] (wake event 24)
- WAKE_STATUS[27] = gp3_ps[0] (wake event 27)
- WAKE_STATUS[30] = dap1_dout (wake event 30)

This register masks which event can cause a wake-up from Deep Sleep mode. It has to be set up before entering Deep Sleep mode. It works in conjunction with the WAKE_LVL register.

Only enabled events at the proper wake_lvl will cause exit from Deep Sleep mode.

PMC Wake-up Event Mask

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	RESET_N: External reset wake enable 0 = DISABLE 1 = ENABLE
30:23	0x0	EVENT_RES: 0 = DISABLE 1 = ENABLE
22:19	0x0	USB_EVENT: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
18	0x0	PWR_INT_N: PWR_INT_N wake enable 0 = DISABLE 1 = ENABLE
17	0x0	KBC: KBC wake enable 0 = DISABLE 1 = ENABLE
16	0x0	RTC: RTC wake enable 0 = DISABLE 1 = ENABLE
15:0	0x0	EVENT: pin 0-15 wake enable 0 = DISABLE 1 = ENABLE

10.9.5 APBDEV_PMC_WAKE_LVL_0

This register sets the active level for the wake event. It will cause an exit from Deep Sleep mode if the input signal level matches the level set in this register and WAKE_MASK is set for the event to 1.

PMC Wake Level

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x7f9fffff (0b01111111100111111111111111111111)

Bit	Reset	Description
31	0x0	RESET_N: External reset wake level (low active) 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
30:23	0xff	EVENT_RES: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
22:19	0x3	USB_EVENT: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
18	0x1	PWR_INT_N: Power interrupt - permanently tied to bit 18 0 = ACTIVE_LOW 1 = ACTIVE_HIGH

Bit	Reset	Description
17	0x1	KBC: KBC wake level 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
16	0x1	RTC: RTC wake level 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
15:0	0xffff	EVENT: Pin 0-15 wake level 0 = ACTIVE_LOW 1 = ACTIVE_HIGH

10.9.6 APBDEV_PMC_WAKE_STATUS_0

This register stores the status of the wake events. The event will be set if the level matches and is not masked.

A write will reset the set wake event.

PMC Wake Status

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	RESET_N: external reset 0 = NOT_SET 1 = SET
30:23	0x0	EVENT_RES: 0 = NOT_SET 1 = SET
22:19	0x0	USB_EVENT: USB wake events 0 = NOT_SET 1 = SET
18	0x0	PWR_INT_N: power interrupt 0 = NOT_SET 1 = SET
17	0x0	KBC: KBC wake 0 = NOT_SET 1 = SET
16	0x0	RTC: RTC wake 0 = NOT_SET 1 = SET
15:0	0x0	EVENT: pin 0-15 wake 0 = NOT_SET 1 = SET

10.9.7 APBDEV_PMC_SW_WAKE_STATUS_0

This register stores status of the wake events. Latching of the events in software wake status is enabled by the PMC_CNTRL register bit LATCHWAKE_EN.

Latching will stop at a 1-to-0 transition on this bit. An event will be set if the level matches. Masking does not affect this register. A write will reset the set wake events.

PMC Software Wake Status

Offset: 0x18 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	RESET_N: External reset 0 = NOT_SET 1 = SET
30:23	0x0	EVENT_RES: 0 = NOT_SET 1 = SET
22:19	0x0	USB_EVENT: USB wake events 0 = NOT_SET 1 = SET
18	0x0	PWR_INT: Power interrupt 0 = NOT_SET 1 = SET
17	0x0	KBC: KBC wake 0 = DISABLE 1 = ENABLE
16	0x0	RTC: RTC wake 0 = DISABLE 1 = ENABLE
15:0	0x0	EVENT: Pin 0-15 wake 0 = DISABLE 1 = ENABLE

10.9.8 APBDEV_PMC_DPD_PADS_ORIDE_0

This register enables overriding values from pinmux with values driven by the KBC (keyboard) or PMC (sys_clk_req, blink).

If a bit is set to 1, the associated I/O will drive the direct value from the KBC or PMC, not the pinmux value. During Deep Sleep mode, the pads with the bit set to 1 will not be in Deep Power Down mode. It will not drive data from mini pad macros stored during sample cycle, but direct data from the KBC or PMC.

Note: This register has to be set before entering Deep Sleep mode. The meaning of fields is no longer significant.

DPD Pads Override

Offset: 0x1c │ Read/Write: R/W │ Reset: 0x00200000 (0bxxxxxx00001000000000000000000000)

Bit	Reset	Description
25	0x0	KBC_ROW10: Override DPD idle state of the gp3_pq[7] pad 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
24	0x0	KBC_ROW9: Override DPD idle state of the gp3_pq[6] pad 0 = DISABLE 1 = ENABLE
23	0x0	KBC_ROW8: Override DPD idle state of the gp3_pq[5] pad 0 = DISABLE 1 = ENABLE
22	0x0	KBC_ROW7: Override DPD idle state of the gp3_pq[4] pad 0 = DISABLE 1 = ENABLE
21	0x1	SYS_CLK_REQ: Override DPD idle state with column with SYS_CLK_REQ output 0 = DISABLE 1 = ENABLE
20	0x0	BLINK: Override DPD idle state with blink output 0 = DISABLE 1 = ENABLE
19	0x0	KBC_ROW6: 0 = DISABLE 1 = ENABLE
18	0x0	KBC_ROW5: 0 = DISABLE 1 = ENABLE
17	0x0	KBC_ROW4: 0 = DISABLE 1 = ENABLE
16	0x0	KBC_ROW3: 0 = DISABLE 1 = ENABLE
15	0x0	KBC_ROW2: 0 = DISABLE 1 = ENABLE
14	0x0	KBC_ROW1: Override DPD idle state of the gp3_pq[3] pad 0 = DISABLE 1 = ENABLE
13	0x0	KBC_ROW0: Override DPD idle state of the gp3_pq[2] pad 0 = DISABLE 1 = ENABLE
12	0x0	KBC_COL12: Override DPD idle state of the gp3_pq[1] pad 0 = DISABLE 1 = ENABLE
11	0x0	KBC_COL11: Override DPD idle state of the gp3_pq[0] pad 0 = DISABLE 1 = ENABLE
10	0x0	KBC_COL10: Override DPD idle state of the gp3_ps[2] pad 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	0x0	KBC_COL9: Override DPD idle state of the gp3_ps[1] pad 0 = DISABLE 1 = ENABLE
8	0x0	KBC_COL8: Override DPD idle state of the gp3_ps[0] pad 0 = DISABLE 1 = ENABLE
7	0x0	KBC_COL7: Override DPD idle state of the gp3_pr[7] pad 0 = DISABLE 1 = ENABLE
6	0x0	KBC_COL6: Override DPD idle state of the gp3_pr[6] pad 0 = DISABLE 1 = ENABLE
5	0x0	KBC_COL5: Override DPD idle state of the gp3_pr[5] pad 0 = DISABLE 1 = ENABLE
4	0x0	KBC_COL4: Override DPD idle state of the gp3_pr[4] pad 0 = DISABLE 1 = ENABLE
3	0x0	KBC_COL3: Override DPD idle state of the gp3_pr[3] pad 0 = DISABLE 1 = ENABLE
2	0x0	KBC_COL2: Override DPD idle state of the gp3_pr[2] pad 0 = DISABLE 1 = ENABLE
1	0x0	KBC_COL1: Override DPD idle state of the gp3_pr[1] pad 0 = DISABLE 1 = ENABLE
0	0x0	KBC_COL0: Override DPD idle state of the gp3_pr[0] pad 0 = DISABLE 1 = ENABLE

10.9.9 APBDEV_PMC_DPD_SAMPLE_0

Setting this register will trigger sampling pads data and direction in which the pad will be driven during Deep Sleep mode.

Before writing to this register, all interfaces going to pads must be set to the ideal "idle" mode, which is expected to be driven by pads when the Tegra 4 chip enters Deep Sleep.

DPS Power Down Sample has to precede Deep Power Down Enable write. The DPD sample should not be deasserted until the transition from Deep Sleep to WB0.

The DPD sample feature works only for GPIO (MPIO) pads, not for brick pads.

Deep Power Down Sample

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ON: Sets the sampling of pads value 0 = DISABLE 1 = ENABLE

10.9.10 APBDEV_PMC_DPD_ENABLE_0

Setting this register will trigger entering Deep Sleep state. It must be preceded by a DPD_SAMPLE write.

Will cause request for shutting down the system clock and the core req power. Puts the PLLs and I/Os in Deep Power Down mode. Will cut off (clamp) all signals going from core power to AO and pads.

If none of the wake-up events is set, only power-on reset can re-enable access to the chip.

After servicing of the wake-up event is completed, the register should be set back to 0 to complete a Deep Sleep cycle.

Deep Power Down Enable

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	TSC_MULT_EN: TSC multiplier enable. 0: DISABLE (i.e., counter runs at oscillator clock and increments by 1 every clock cycle). 1: ENABLE (i.e., counter runs at 32 kHz and increments by TSC_MULT_VALUE every clock cycle). 0 = DISABLE 1 = ENABLE
0	0x0	ON: Sets sampling of pads value 0 = DISABLE 1 = ENABLE

10.9.11 APBDEV_PMC_PWRGATE_TIMER_OFF_0

Specifies the number of APB cycles after which the rail line goes off (turns the power to part of power-gated partition). Each rail controls part of the powered partition. This register should be set before the write to Power Gate Toggle. Shared between all power partitions.

Power Gate Timer Off Register

Offset: 0x28 | Read/Write: R/W | Reset: 0xeca97531 (0b11101100101010010111010100110001)

Bit	Reset	Description
31:28	0xe	RAIL7: Timer value for rail 7
27:24	0xc	RAIL6: Timer value for rail 6
23:20	0xa	RAIL5: Timer value for rail 5
19:16	0x9	RAIL4: Timer value for rail 4
15:12	0x7	RAIL3: Timer value for rail 3

Bit	Reset	Description
11:8	0x5	RAIL2: Timer value for rail 2
7:4	0x3	RAIL1: Timer value for rail 1
3:0	0x1	RAIL0: Timer value for rail 0

10.9.12 APBDEV_PMC_CLAMP_STATUS_0

This register is kept for backward code compatibility; the timer is based on PWRGATE_TIMER_OFF only.

Offset: 0x2c │ Read/Write: RO │ Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
22	X	XUSBC: clamp status of XUSBC partition 0 = DISABLE 1 = ENABLE
21	X	XUSBB: clamp status of XUSBB partition 0 = DISABLE 1 = ENABLE
20	X	XUSBA: clamp status of XUSBA partition 0 = DISABLE 1 = ENABLE
19	X	DISB: clamp status of DISB partition 0 = DISABLE 1 = ENABLE
18	X	DIS: clamp status of DIS partition 0 = DISABLE 1 = ENABLE
16	X	C1NC: clamp status of cluster1 nENABLE CPU partition 0 = DISABLE 1 = ENABLE
15	X	C0NC: clamp status of cluster0 nENABLE CPU partition 0 = DISABLE 1 = ENABLE
14	X	CE0: clamp status of CE0 partition 0 = DISABLE 1 = ENABLE
12	X	CELP: clamp status of CELP partition 0 = DISABLE 1 = ENABLE
11	X	CE3: clamp status of CE3 partition 0 = DISABLE 1 = ENABLE
10	X	CE2: clamp status of CE2 partition 0 = DISABLE 1 = ENABLE
9	X	CE1: clamp status of CE1 partition 0 = DISABLE 1 = ENABLE
7	X	HEG: clamp status of HEG partition 0 = DISABLE 1 = ENABLE
6	X	MPE: Clamp status of MPE partition 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	X	VDE: Clamp status of VDE partition 0 = DISABLE 1 = ENABLE
2	X	VE: clamp status of VE partition 0 = DISABLE 1 = ENABLE
1	X	TD: clamp status of TD partition 0 = DISABLE 1 = ENABLE
0	X	CRAIL: clamp status of CPU Rail 0 = DISABLE 1 = ENABLE

10.9.13 APBDEV_PMC_PWRGATE_TOGGLE_0

Write to this register will turn power on/off to the specified power-gated partition. Only one partition is turned on/off at a time.

PWRGATE_STATUS should be read to determine the state of the partition before writing to this register.

Turning the partition off will cause automatic clamping of all signals generated by the power-gated partition being turned off. Before the partition is turned off, all clocks to the partition should be stopped, and the reset to the partition should be asserted.

Turning the partition on will not remove clamping. Clamping is removed only after a REMOVE_CLAMPING_CMD write.

The user must allow a minimum 20 APB clock cycles between consecutive partition Power-Gate Toggle requests.

The role of the START bit has changed from prior Tegra devices. The START bit is cleared by hardware when the PMC accepts the request to power-gate or unpower-gate the partition. So in order to power-gate/unpower-gate a partition, software needs to do the following:

- Check to see if the partition is already in the correct state, by looking at the PWRGATE_STATUS register.
- If the partition is not in the correct state, software reads the PWRGATE_TOGGLE register to see if the START bit is 0.
- If the START bit is not 0, software polls until the start bit is set to 0.
- Then program the PWRGATE_TOGGLE register with the START bit set to 1 and choose the required partition to be power-gated.
- Ideally, software can poll to check the START bit going back to 0, which indicates that the PMC has accepted the request, and then poll the STATUS register to make sure the required partition is power-gated/unpower-gated.

Power Gate Toggle

Offset: 0x30 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxx00000)

Bit	Reset	Description
8	0x0	START: Start power down/up 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4:0	0x0	PARTID: ID of the partition to be toggled 0 = CRAIL 1 = TD 2 = Video Encode 4 = Video Decode 5 = L2 Cache 6 = MPEG Encode 7 = HEG 9 = CE1 10 = CE2 11 = CE3 12 = CELP 14 = CE0 15 = C0NC 16 = C1NC 18 = DIS 19 = DISB 20 = XUSBA 21 = XUSBB 22 = XUSBC

10.9.14 APBDEV_PMC_REMOVE_CLAMPING_CMD_0

This is a bitmap with one bit per power partition controller by the PMC.

When written to 1b, the PMC removes the clamp signals to the corresponding partition. If the partition is not powered on, the register write will be ignored.

The bit is automatically reset to 0b when the clamping has been removed. Software is responsible for writing to this register at the correct time, that is, when the clocks are started but the blocks in that partition are still held in reset.

Remove Clamping

Offset: 0x34 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxx00000x000x0000x000x0000)

Bit	Reset	Description
22	0x0	XUSBC: Remove clamping from XUSBC partition 0 = DISABLE 1 = ENABLE
21	0x0	XUSBB: Remove clamping from XUSBB partition 0 = DISABLE 1 = ENABLE
20	0x0	XUSBA: Remove clamping from XUSBA partition 0 = DISABLE 1 = ENABLE
19	0x0	DISB: Remove clamping from DISB partition 0 = DISABLE 1 = ENABLE
18	0x0	DIS: Remove clamping from DIS partition 0 = DISABLE 1 = ENABLE
16	0x0	C1NC: Remove clamping from cluster 1 non-CPU partition 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
15	0x0	C0NC: Remove clamping from cluster 0 non-CPU partition 0 = DISABLE 1 = ENABLE
14	0x0	CE0: Remove clamping from CE0 partition 0 = DISABLE 1 = ENABLE
12	0x0	A9LP: Remove clamping from CELP partition 0 = DISABLE 1 = ENABLE
11	0x0	CE3: Remove clamping from CE3 partition 0 = DISABLE 1 = ENABLE
10	0x0	CE2: Remove clamping from CE2 partition 0 = DISABLE 1 = ENABLE
9	0x0	CE1: Remove clamping from CE1 partition 0 = DISABLE 1 = ENABLE
7	0x0	HEG: Remove clamping from HEG partition 0 = DISABLE 1 = ENABLE
6	0x0	MPE: Remove clamping from MPE_CACHE partition 0 = DISABLE 1 = ENABLE
5	0x0	L2C: Remove clamping from L2_CACHE partition 0 = DISABLE 1 = ENABLE
3	0x0	VDE: Remove clamping from VDE partition 0 = DISABLE 1 = ENABLE
2	0x0	VE: Remove clamping from VE partition 0 = DISABLE 1 = ENABLE
1	0x0	TD: Remove clamping from TD partition 0 = DISABLE 1 = ENABLE
0	0x0	CRAIL: Remove clamping from CPU Rail domain 0 = DISABLE 1 = ENABLE

10.9.15 APBDEV_PMC_PWRGATE_STATUS_0

This read-only register displays the status of the power partitions. This register should be read before writing to PWRGATE_TOGGLE.

Power Gate Status

Offset: 0x38 │ Read/Write: RO │ Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
22	X	XUSBC: Status of XUSBC partition 0 = OFF 1 = ON
21	X	XUSBB: Status of XUSBB partition 0 = OFF 1 = ON
20	X	XUSBA: Status of XUSBA partition 0 = OFF 1 = ON
19	X	DISB: Status of DISB partition 0 = OFF 1 = ON
18	X	DIS: Status of DIS partition 0 = OFF 1 = ON
16	X	C1NC: Status of cluster1 non-CPU partition 0 = OFF 1 = ON
15	X	C0NC: Status of cluster0 non-CPU partition 0 = OFF 1 = ON
14	X	CE0: Status of CE0 partition 0 = OFF 1 = ON
12	X	CELP: Status of CELP partition 0 = OFF 1 = ON
11	X	CE3: Status of CE3 partition 0 = OFF 1 = ON
10	X	CE2: Status of CE2 partition 0 = OFF 1 = ON
9	X	CE1: Status of CE1 partition 0 = OFF 1 = ON
7	X	HEG: Status of HEG partition 0 = OFF 1 = ON
6	X	MPE: Status of MPE partition 0 = OFF 1 = ON
5	X	L2C: Status of L2C partition 0 = OFF 1 = ON

Bit	Reset	Description
4	X	VDE: Status of VDE partition 0 = OFF 1 = ON
2	X	VE: Status of VE partition 0 = OFF 1 = ON
1	X	TD: Status of TD partition 0 = OFF 1 = ON
0	X	CRAIL: Status of CPU Rail 0 = OFF 1 = ON

10.9.16 APBDEV_PMC_PWRGOOD_TIMER_0

This register programs the length of the wake-up reset, asserted after wake-up from Deep Sleep. This register should be set before entering Deep Sleep mode.

Programmed values depend on the properties of the PMIC.

Timer value x 30.51 µs = reset pulse length

Power Good Timer

Offset: 0x3c │ Read/Write: R/W │ Reset: 0x0000007f (0bxxxxxxxxxxxxxxxx0000000001111111)

Bit	Reset	Description
15:8	0x0	DATA: Crystal (Xtal) timer * 32
7:0	0x7f	PMU_TIM: PMIC timer * 32

10.9.17 APBDEV_PMC_BLINK_TIMER_0

Will output value to pad only if the PMC Control register has BLINK_EN set and DPD_PADS_ORIDE has blink bit set.

Setting bit 15 to 1 will output 32 kHz clock (the registers above still have to be set)

- On time DATA_ON << 2 x 30.51 µs
- Off time DATA_OFF << 2 x 30.51 µs.

Blinker Timer for External Blinker

Offset: 0x40 │ Read/Write: R/W │ Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:16	0xffff	DATA_OFF: Time off
15	0x1	FORCE_BLINK: if 0 32 kHz clock
14:0	0x7fff	DATA_ON: Time on

10.9.18 APBDEV_PMC_NO_IOPOWER_0

This register controls the power rails for each I/O level. It can only be set when a specific power rail is off. Setting it otherwise will damage the pad.

No I/O Power Register

Offset: 0x44 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000x00x0x000000)

Bit	Reset	Description
16	0x0	MEM_COMP:MEM0 ADDR1 (comp cell I/Os) 0 = DISABLE 1 = ENABLE
15	0x0	HV: rail HV I/Os 0 = DISABLE 1 = ENABLE
14	0x0	SDMMC4: rail SDMMC4 I/Os 0 = DISABLE 1 = ENABLE
13	0x0	SDMMC3: rail SDMMC3 I/Os 0 = DISABLE 1 = ENABLE
12	0x0	SDMMC1: rail SDMMC1 I/Os 0 = DISABLE 1 = ENABLE
10	0x0	CAM: Cam rail I/Os 0 = DISABLE 1 = ENABLE
9	0x0	MIPI: MIPI rail I/Os 0 = DISABLE 1 = ENABLE
7	0x0	MEM: rail memory I/Os 0 = DISABLE 1 = ENABLE
5	0x0	AUDIO: rail I2S I/Os 0 = DISABLE 1 = ENABLE
4	0x0	VI: Defunct DVI I/Os 0 = DISABLE 1 = ENABLE
3	0x0	BB: rail DLCD I/Os 0 = DISABLE 1 = ENABLE
2	0x0	UART: rail DBG I/Os 0 = DISABLE 1 = ENABLE
1	0x0	NAND: rail AT3 I/Os 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	SYS: rail AO I/Os 0 = DISABLE 1 = ENABLE

10.9.19 APBDEV_PMC_PWR_DET_0

Active high, sets power detection for nine power rails only. - MIPI does not have power detect.

The proper sequence for turning on power detects:

- Write 1 to PWR_DET register for fields requiring power detect
- Allow for 3 µs delay (in APB clock cycles)
- Write 1 to PWR_DET_LATCH

For turning PWR_DET off:

- Write 0 to PWR_DET_LATCH
- No delay is necessary
- Write 0 to PWR_DET

Power Detect

Offset: 0x48 │ Read/Write: R/W │ Reset: 0x0000b42f (0bxxxxxxxxxxxxxxxx1x11x1xxxx1x1111)

Bit	Reset	Description
15	0x1	HV:HV rail 0 = DISABLE 1 = ENABLE
13	0x1	SDMMC3: rail SDMMC3 I/Os 0 = DISABLE 1 = ENABLE
12	0x1	SDMMC1: rail SDMMC1 I/Os 0 = DISABLE 1 = ENABLE
10	0x1	CAM: rail Cam I/Os 0 = DISABLE 1 = ENABLE
5	0x1	AUDIO: rail I2S I/Os 0 = DISABLE 1 = ENABLE
3	0x1	BB:rail DLCD I/Os 0 = DISABLE 1 = ENABLE
2	0x1	UART: rail DBG I/Os 0 = DISABLE 1 = ENABLE
1	0x1	NAND: rail GMI I/Os 0 = DISABLE 1 = ENABLE
0	0x1	SYS: rail AO I/Os 0 = DISABLE 1 = ENABLE

10.9.20 APBDEV_PMC_PWR_DET_LATCH_0

Latches power detect for power rails enabled by Power Detect register.

Power Detect Latch

Offset: 0x4c │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	LATCH: power detect latch, latches value as long set to 1 0 = DISABLE 1 = ENABLE

10.9.21 APBDEV_PMC_SCRATCH0_0

Scratch Register

Scratch registers for restoring context after wake-up. On a cold power up, the content of register 0 will be reset to 0x0.

Offset: 0x50 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH0: General-purpose register storage

10.9.22 APBDEV_PMC_SCRATCH1_0

Scratch Register

Offset: 0x54 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH1: General-purpose register storage

10.9.23 APBDEV_PMC_SCRATCH2_0

Scratch Register

Offset: 0x58 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH2: General-purpose register storage

10.9.24 APBDEV_PMC_SCRATCH3_0

Scratch Register

Offset: 0x5c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH3: General-purpose register storage

10.9.25 APBDEV_PMC_SCRATCH4_0

Scratch Register

Offset: 0x60 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH4: General-purpose register storage

10.9.26 APBDEV_PMC_SCRATCH5_0

Scratch Register

Offset: 0x64 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH5: General-purpose register storage

10.9.27 APBDEV_PMC_SCRATCH6_0

Scratch Register

Offset: 0x68 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH6: General-purpose register storage

10.9.28 APBDEV_PMC_SCRATCH7_0

Scratch Register

Offset: 0x6c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH7: General-purpose register storage

10.9.29 APBDEV_PMC_SCRATCH8_0

Scratch Register

Offset: 0x70 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH8: General-purpose register storage

10.9.30 APBDEV_PMC_SCRATCH9_0

Scratch Register

Offset: 0x74 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH9: General-purpose register storage

10.9.31 APBDEV_PMC_SCRATCH10_0

Scratch Register

Offset: 0x78 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH10: General-purpose register storage

10.9.32 APBDEV_PMC_SCRATCH11_0

Scratch Register

Offset: 0x7c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH11: General-purpose register storage

10.9.33 APBDEV_PMC_SCRATCH12_0

Scratch Register

Offset: 0x80 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH12: General-purpose register storage

10.9.34 APBDEV_PMC_SCRATCH13_0

Scratch Register

Offset: 0x84 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH13: General-purpose register storage

10.9.35 APBDEV_PMC_SCRATCH14_0

Scratch Register

Offset: 0x88 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH14: General-purpose register storage

10.9.36 APBDEV_PMC_SCRATCH15_0

Scratch Register

Offset: 0x8c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH15: General-purpose register storage

10.9.37 APBDEV_PMC_SCRATCH16_0

Scratch Register

Offset: 0x90 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH16: General-purpose register storage

10.9.38 APBDEV_PMC_SCRATCH17_0

Scratch Register

Offset: 0x94 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH17: General-purpose register storage

10.9.39 APBDEV_PMC_SCRATCH18_0

Scratch Register

Offset: 0x98 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH18: General-purpose register storage

10.9.40 APBDEV_PMC_SCRATCH19_0

Scratch Register

Offset: 0x9c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH19: General-purpose register storage

10.9.41 APBDEV_PMC_SCRATCH20_0

Scratch Register

Offset: 0xa0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH20: General-purpose register storage

10.9.42 APBDEV_PMC_SCRATCH21_0

Scratch Register

Offset: 0xa4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH21: General-purpose register storage

10.9.43 APBDEV_PMC_SCRATCH22_0

Scratch Register

Offset: 0xa8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH22: General-purpose register storage

10.9.44 APBDEV_PMC_SCRATCH23_0

Scratch Register

Offset: 0xac | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH23: General-purpose register storage

10.9.45 APBDEV_PMC_SECURE_SCRATCH0_0

Secure Scratch Register

Offset: 0xb0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH0

10.9.46 APBDEV_PMC_SECURE_SCRATCH1_0

Secure Scratch Register

Offset: 0xb4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH1

10.9.47 APBDEV_PMC_SECURE_SCRATCH2_0

Secure Scratch Register

Offset: 0xb8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH2

10.9.48 APBDEV_PMC_SECURE_SCRATCH3_0

Secure Scratch Register

Offset: 0xbc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH3

10.9.49 APBDEV_PMC_SECURE_SCRATCH4_0

Secure Scratch Register

Offset: 0xc0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH4

10.9.50 APBDEV_PMC_SECURE_SCRATCH5_0

Secure Scratch Register

Offset: 0xc4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH5

10.9.51 APBDEV_PMC_CPUPWRGOOD_TIMER_0

CPU Power Good Timer. The behavior for the DATA timers 0 to 2 is the same (i.e., 2 cycles of delay).

Offset: 0xc8 | Read/Write: R/W | Reset: 0x0000ffff (0b00000000000000001111111111111111)

Bit	Reset	Description
31:0	0xffff	DATA: timer data

10.9.52 APBDEV_PMC_CPUPWROFF_TIMER_0

CPU Power Off Timer

Used for On-to-Off transitions.

Note: The value of this timer must be greater than 1.

Offset: 0xcc | Read/Write: R/W | Reset: 0x0000ffff (0b00000000000000001111111111111111)

Bit	Reset	Description
31:0	0xffff	DATA: timer data

10.9.53 APBDEV_PMC_PG_MASK_0

Offset: 0xd0 | Read/Write: R/W | Reset: 0x00ffffff (0bxxxxxxxx111111111111111111111111)

Bit	Reset	Description
23:16	0xff	VD: Mask VDE rail
15:8	0xff	VE: Mask VE rail
7:0	0xff	TD: Mask TD rail

10.9.54 APBDEV_PMC_PG_MASK_1_0

Offset: 0xd4 | Read/Write: R/W | Reset: 0x00ffff01 (0bxxxxxxxx1111111111111111xxxxxxx1)

Bit	Reset	Description
23:16	0xff	HEG: MASK HEG rail
15:8	0xff	MPE: MASK MPE rail
0	0x1	L2C: MASK L2C rail

10.9.55 APBDEV_PMC_AUTO_WAKE_LVL_0

Note: This register should not be programmed.

The wake levels are snapped just before entering DPD by default.

Offset: 0xd8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	SMPL: Causes PMC to sample the wake pads 0 = DISABLE 1 = ENABLE

10.9.56 APBDEV_PMC_AUTO_WAKE_LVL_MASK_0

This register is used by software to enable sampling of the wake pads before entering Deep Sleep.

Setting a '1' causes the associated pad to be sampled, and the value transferred to the WAKE_LVL register.

Offset: 0xdc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE

10.9.57 APBDEV_PMC_WAKE_DELAY_0

Offset: 0xe0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	VALUE

10.9.58 APBDEV_PMC_PWR_DET_VAL_0

This register is used to override the power-detect cells and manually set the values (in the unlikely case the power-detect cells are broken). A write to this register also causes the values from the power-detect cells to be captured and which can be subsequently read out.

Offset: 0xe4 │ Read/Write: R/W │ Reset: 0x0000b42f (0bxxxxxxxxxxxxxxxx1x11x1xxxx1x1111)

Bit	Reset	Description
15	0x1	HV:rail HV I/Os 0 = ENABLE 1 = DISABLE
13	0x1	SDMMC3: rail SDMMC3 I/Os 0 = ENABLE 1 = DISABLE
12	0x1	SDMMC1: rail SDMMC1 I/Os 0 = ENABLE 1 = DISABLE
10	0x1	CAM: rail cam I/Os 0 = ENABLE 1 = DISABLE
5	0x1	AUDIO: rail I2S I/Os 0 = ENABLE 1 = DISABLE
3	0x1	BB: rail DLCD I/Os 0 = ENABLE 1 = DISABLE
2	0x1	UART: rail DBG I/Os 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
1	0x1	NAND: rail AT3 I/Os 0 = ENABLE 1 = DISABLE
0	0x1	SYS: rail AO I/Os 0 = ENABLE 1 = DISABLE

10.9.59 APBDEV_PMC_DDR_PWR_0

This register is used to program the "E_18V" pin of the DDR pads.

Offset: 0xe8 │ Read/Write: R/W │ Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11)

Bit	Reset	Description
1	0x1	EMMC: GMI pins 0 = E_12V 1 = E_18V
0	0x1	VAL: DVI pins 0 = E_12V 1 = E_18V

10.9.60 APBDEV_PMC_USB_DEBOUNCE_DEL_0

Determines number of 32 kHz clock cycles to debounce USB signal events.

Note: The programmed debounce values must be greater than 1.

Reset values are for Cold Hardware Reset only.

Offset: 0xec │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:20	0x0	UHSIC_LINE_DEB_CNT: Number of 32 kHz debounce cycles for UHSIC port 0
19:16	0x0	UTMIP_LINE_DEB_CNT: Number of 32 kHz debounce cycles for UTMIP port 0
15:0	0x0	VAL: Debounce period for ID and VBUS events on all USB ports

10.9.61 APBDEV_PMC_USB_AO_0

Power downs for various USB features controlled by the PMC.

Each UTMIP has the following power downs for features that can lead to wake-up events.

- USBOP_VAL_PD: Power down for D+ value detector (I/O pad)
- USBON_VAL_PD: Power down for D- value detector (I/O pad)
- ID_PD: Power down for ID detector (bias pad)
- VBUS_WAKEUP_PD: Power down for VBUS detector (bias pad)

Each UHSIC port has the following power downs for features that can lead to wake-up events:

- STROBE_VAL_PD: Power down for STROBE VALUE detector
- DATA_VAL_PD: Power down for DATA VALUE detector

Note: These HSIC pins have not been added to the pad yet.

Line Debounce Periods for UTMIP and HSIC PORTS

Offset: 0xf0 | Read/Write: R/W | Reset: 0x000fffff (0bxxxxxxxxxxxx11111111111111111111)

Bit	Reset	Description
19:18	0x3	UHSIC_RESERVED_P1: 2 bits reserved for UHSIC P1
17	0x1	DATA_VAL_PD_P1: Power Down D- DATA_VAL receiver for UHSIC P1
16	0x1	STROBE_VAL_PD_P1: Power Down D+ STROBE_VAL receiver for UHSIC P1
15:14	0x3	UHSIC_RESERVED_P0: 2 bits reserved for UHSIC P1
13	0x1	DATA_VAL_PD_P0: Power Down D- DATA_VAL receiver for UHSIC P0
12	0x1	STROBE_VAL_PD_P0: Power Down D+ STROBE_VAL receiver for UHSIC P0
11	0x1	ID_PD_P2: Power Down ID Wake up for UTMIP
10	0x1	VBUS_WAKEUP_PD_P2: Power Down Vbus Wake Up for UTMIP P2
9	0x1	USBON_VAL_PD_P2: Power Down D- USBOP_VAL receiver for UTMIP P0
8	0x1	USBOP_VAL_PD_P2: Power Down D+ USBOP_VAL receiver for UTMIP P0
7	0x1	ID_PD_P1: Power Down ID Wake up for UTMIP P1
6	0x1	VBUS_WAKEUP_PD_P1: Power Down Vbus Wake Up for UTMIP P1
5	0x1	USBON_VAL_PD_P1: Power Down D- USBOP_VAL receiver for UTMIP P0
4	0x1	USBOP_VAL_PD_P1: Power Down D+ USBOP_VAL receiver for UTMIP P0
3	0x1	ID_PD_P0: Power Down ID Wake up for UTMIP
2	0x1	VBUS_WAKEUP_PD_P0: Power Down Vbus Wake Up for UTMIP P0
1	0x1	USBON_VAL_PD_P0: Power Down D- USBOP_VAL receiver for UTMIP P0
0	0x1	USBOP_VAL_PD_P0: Power Down D+ USBOP_VAL receiver for UTMIP P0

10.9.62 APBDEV_PMC_CRYPTO_OP_0

A complete solution requires a "semi-sticky" bit in the always-on domain. The Boot ROM would clear this semi-sticky bit for cold boots, but use its value to propagate the crypto-disable flag across LP0. (The Boot ROM always requires crypto functionality, so a pure hardware solution probably isn't reasonable.)

The Boot ROM would be able to clear (to zero) and read the sticky bit, but outside the Boot ROM users would only be able to set (to one) and read the sticky bit. On cold boot, the Boot ROM would always clear the stick bit. On WB0, the Boot ROM would read the sticky bit and copy it's setting to the crypto disable flag.

Crypto Engine Disable Sticky Bit

Offset: 0xf4 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	VAL: Disabled by default 0 = ENABLE 1 = DISABLE

10.9.63 APBDEV_PMC_PLLP_WB0_OVERRIDE_0

Master control for all WB0 PLL overrides.

Offset: 0xf8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12	0x0	PLLM_ENABLE: 1 = enable PLLM, 0 = disable PLLM
11	0x0	PLLM_OVERRIDE_ENABLE: 1 = override CAR PLLM setting, 0 = no override. PLLM WB to programmable frequency.
10	0x0	PLLU_ENABLE: 1 = enable PLLU, 0 = disable PLLU.
9	0x0	PLLU_OVERRIDE_ENABLE: 1 = override CAR PLLU setting, 0 = no override. PLLU WB to fixed frequency
8	0x0	OSC_OVERRIDE_ENABLE: 1 = override CAR OSC setting, 0 = no override. Controls OSC_FREQ and PLL_REF_DIV
7:6	0x0	PLL_REF_DIV: PLL reference clock divide for all PLLs. 00 = /1, 01 = /2, 10 = /4, 11 = reserve.
5:2	0x0	OSC_FREQ: Oscillator frequency for shared PLL reference 0000 = 13MHz, 0100 = 19.2MHz, 1000 = 12MHz, 1100 = 26MHz. 0001 = 16.8MHz, 0101 = 38.4MHz*, 1001 = 48.0MHz
1	0x0	PLLP_ENABLE: 1 = enable PLLP, 0 = disable PLLP
0	0x0	PLLP_OVERRIDE_ENABLE: 1 = override CAR PLLP setting, 0 = no override. PLLP WB to fixed frequency.

10.9.64 APBDEV_PMC_SCRATCH24_0

Scratch Register

Offset: 0xfc │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH24: General-purpose register storage

10.9.65 APBDEV_PMC_SCRATCH25_0

Scratch Register

Offset: 0x100 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH25: General-purpose register storage

10.9.66 APBDEV_PMC_SCRATCH26_0

Scratch Register

Offset: 0x104 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH26: General-purpose register storage

10.9.67 APBDEV_PMC_SCRATCH27_0

Scratch Register

Offset: 0x108 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH27: General-purpose register storage

10.9.68 APBDEV_PMC_SCRATCH28_0

Scratch Register

Offset: 0x10c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH28: General-purpose register storage

10.9.69 APBDEV_PMC_SCRATCH29_0

Scratch Register

Offset: 0x110 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH29: General-purpose register storage

10.9.70 APBDEV_PMC_SCRATCH30_0

Scratch Register

Offset: 0x114 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH30: General-purpose register storage

10.9.71 APBDEV_PMC_SCRATCH31_0

Scratch Register

Offset: 0x118 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH31: General-purpose register storage

10.9.72 APBDEV_PMC_SCRATCH32_0

Scratch Register

Offset: 0x11c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH32: General-purpose register storage

10.9.73 APBDEV_PMC_SCRATCH33_0

Scratch Register

Offset: 0x120 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH33: General-purpose register storage

10.9.74 APBDEV_PMC_SCRATCH34_0

Scratch Register

Offset: 0x124 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH34: General-purpose register storage

10.9.75 APBDEV_PMC_SCRATCH35_0

Scratch Register

Offset: 0x128 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH35: General-purpose register storage

10.9.76 APBDEV_PMC_SCRATCH36_0

Scratch Register

Offset: 0x12c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH36: General-purpose register storage

10.9.77 APBDEV_PMC_SCRATCH37_0

Scratch Register

Offset: 0x130 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH37: General-purpose register storage

10.9.78 APBDEV_PMC_SCRATCH38_0

Scratch Register

Offset: 0x134 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH38: General-purpose register storage

10.9.79 APBDEV_PMC_SCRATCH39_0

Scratch Register

Offset: 0x138 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH39: General-purpose register storage

10.9.80 APBDEV_PMC_SCRATCH40_0

Scratch Register

Offset: 0x13c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH40: General-purpose register storage

10.9.81 APBDEV_PMC_SCRATCH41_0

Scratch Register

Offset: 0x140 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH41: General-purpose register storage

10.9.82 APBDEV_PMC_SCRATCH42_0

Scratch Register

Offset: 0x144 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH42: General-purpose register storage

10.9.83 APBDEV_PMC_BONDOUT_MIRROR0_0

Secure Scratch Register

Offset: 0x148 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR0

10.9.84 APBDEV_PMC_BONDOUT_MIRROR1_0

Secure Scratch Register

Offset: 0x14c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR1

10.9.85 APBDEV_PMC_BONDOUT_MIRROR2_0

Secure Scratch Register

Offset: 0x150 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR2

10.9.86 APBDEV_PMC_SYS_33V_EN_0

Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	VAL: 1 = 3.3V 0 = 1.8V

10.9.87 APBDEV_PMC_BONDOUT_MIRROR_ACCESS_0

On separate reset (same as the CAR) disables access (read/write to secure scratch registers)

Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	BREAD: disable read from bondout secure registers 0 = OFF 1 = ON
0	0x0	BWRITE: disable write to bondout secure registers 0 = OFF 1 = ON

10.9.88 APBDEV_PMC_GATE_0

This register is used for software controlled synchronization between APB domain and the 32 kHz domain. Software is expected to set the GAKE_WAKE/GATE_DBNS field high to cut the 32 kHz gated clock before updating WAKE_LVL, AUTO_WAKE_LVL, WAKE_MASK and USB_DEBOUNCE_DEL registers. After these registers have been written the GATE_WAKE/GATE_DBNS bit should be written back to '0' to enable the 32 kHz gated clock.

This gates the 32 kHz clock to just the flops that store the above fields.

Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	GATE_WAKE: 0 = OFF 1 = ON
0	0x0	GATE_DBNS: 0 = OFF 1 = ON

10.9.89 APBDEV_PMC_WAKE2_MASK_0

Auto-wake is not present for wake2 events in Tegra 4 devices.

The APBDEV_PMC_WAKE2 registers handle wake events whose number exceeds 32. For wave events whose number is less than or equal to 32, refer to the APBDEV_PMC_WAKE registers above.

Wake2 alignments for masks, level, and status:

- WAKE2_*[0] = ulpi_data3 (wake event 32)
- WAKE2_*[1] = gmi_cs0_n (wake event 33)
- WAKE2_*[2] = gmi_cs4_n (wake event 34)
- WAKE2_*[3] = gmi_cs7_n (wake event 35)
- WAKE2_*[12]= i2c1_sda (wake event 44)
- WAKE2_*[13]= gpio_pbb6 (wake event 45)
- WAKE2_*[14]= pwr_i2c_sda (wake event 46)
- WAKE2_*[15]= gp3_pt[6] (wake event 47)
- WAKE2_*[16]= cam_i2c_sda (wake event 48)
- WAKE2_*[17]= gp3_pr[7] (wake event 49)
- WAKE2_*[18]= gp3_pr[4] (wake event 50)
- WAKE2_*[19]= gp3_pq[0] (wake event 51)
- WAKE2_*[20]= cec (wake event 52)
- WAKE2_*[21]= cam_i2c_scl (wake event 53)
- WAKE2_*[22]= gp3_pq[5] (wake event 54)
- WAKE2_*[23]= uc3_cts (wake event 55)
- WAKE2_*[24]= gp3_pv[2] (wake event 56)
- WAKE2_*[25]= spdif_in (wake event 57)
- WAKE2_*[26] = wake2pmc_xusb_system_wakeup (wake event 58)

Offset: 0x160 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxx00000000000000000000000000)

Bit	Reset	Description
26:12	0x0	EVENT_REST_1: 0 = DISABLE 1 = ENABLE
11:7	0x0	EVENT_REST: 0 = DISABLE 1 = ENABLE
6:5	0x0	USB_EVENT: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
4:0	0x0	EVENT: 0 = DISABLE 1 = ENABLE

10.9.90 APBDEV_PMC_WAKE2_LVL_0

This register sets the active level for a wake event. It causes an exit from the Deep Sleep state if the input signal level matches the level set in this register and if WAKE2_MASK is set for the event to 1. A level is not needed for 4 line wakeup events; the level is always 1.

PMC Wake Level

Offset: 0x164 │ Read/Write: R/W │ Reset: 0x01ffffe7 (0b00000001111111111111111111100111)

Bit	Reset	Description
31	0x0	ALLOW_PULSE_WAKE: Allows pulse wakes in case the USB wants to wake on pulse; otherwise it is a level wake (the signal must last longer than 1 ms)
30:27	0x0	FUTURE_CNTL: Reserved
26:12	0x1fff	EVENT_REST_1: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
11:7	0x1f	EVENT_REST: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
6:5	0x3	USB_EVENT: 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
4:0	0x7	EVENT: Pin 0-2 wake level 0 = ACTIVE_LOW 1 = ACTIVE_HIGH

10.9.91 APBDEV_PMC_WAKE2_STATUS_0

This register stores status of the wake events. An event is set if the level matches and is not masked.

Writes will reset the set wake event.

PMC Wake Status

Offset: 0x168 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxx00000000000000000000000000)

Bit	Reset	Description
26:12	0x0	EVENT_REST_1: 0 = NOT_SET 1 = SET
11:7	0x0	EVENT_REST: Internally set USB events 0 = NOT_SET 1 = SET
6:5	0x0	USB_EVENT: USB wake events 0 = NOT_SET 1 = SET
4:0	0x0	EVENT: Pin 0-2 wake enable 0 = NOT_SET 1 = SET

10.9.92 APBDEV_PMC_SW_WAKE2_STATUS_0

This register stores the status of the wake events. Latching of the events in software wake status is enabled by the PMC_CNTRL register bit LATCHWAKE_EN. Latching will stop at a 1-to-0 transition on this bit.

An event is set if the level matches. Masking does not affect this register. Writes will reset the set wake events.

PMC Software Wake Status

Offset: 0x16c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxx00000000000000000000000000)

Bit	Reset	Description
26:12	0x0	EVENT_REST_1: 0 = NOT_SET 1 = SET
11:7	0x0	EVENT_REST
6:5	0x0	USB_EVENT: USB wake events 0 = NOT_SET 1 = SET
4:0	0x0	EVENT: pin 0-2 wake 0 = DISABLE 1 = ENABLE

10.9.93 APBDEV_PMC_AUTO_WAKE2_LVL_MASK_0

Auto-wake is not present for wake2 events in Tegra 4 devices.

Offset: 0x170 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxx00000000000000000000000000)

Bit	Reset	Description
26:12	0x0	EVENT_REST_1
11:7	0x0	EVENT_REST
6:0	0x0	VALUE

10.9.94 APBDEV_PMC_PG_MASK_2_0

Power-Gate mask registers for power-gated fields. Need 32 bits for CPU.

Offset: 0x174 │ Read/Write: R/W │ Reset: 0x0000ffff (0bxxxxxxxxxxxxxxxx1111111111111111)

Bit	Reset	Description
15:8	0xff	CELP: Mask CELP rail
7:0	0xff	TD2: Mask TD2 rail - Defunct

10.9.95 APBDEV_PMC_PG_MASK_CE1_0

Offset: 0x178 │ Read/Write: R/W │ Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	MASK: Mask CE1 rail

10.9.96 APBDEV_PMC_PG_MASK_CE2_0

Offset: 0x17c │ Read/Write: R/W │ Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	MASK: Mask CE1 rail

10.9.97 APBDEV_PMC_PG_MASK_CE3_0

Offset: 0x180 | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	MASK: Mask CE1 rail

10.9.98 APBDEV_PMC_PWRGATE_TIMER_CE_0_0

Separate power_gate_off, on, for CE counters - 4 bits each.

Offset: 0x184 | Read/Write: R/W | Reset: 0xfdb97531 (0b11111101101110010111010100110001)

Bit	Reset	Description
31:28	0xf	RAIL7: Timer Value for rail 7
27:24	0xd	RAIL6: Timer Value for rail 6
23:20	0xb	RAIL5: Timer Value for rail 5
19:16	0x9	RAIL4: Timer Value for rail 4
15:12	0x7	RAIL3: Timer Value for rail 3
11:8	0x5	RAIL2: Timer Value for rail 2
7:4	0x3	RAIL1: Timer Value for rail 1
3:0	0x1	RAIL0: Timer Value for rail 0

10.9.99 APBDEV_PMC_PWRGATE_TIMER_CE_1_0

PWRGATE_TIMER CE_1 removed but kept for backward compatibility.

Offset: 0x188 | Read/Write: R/W | Reset: 0x1240e30a (0bxx0100100100000011100011000010 10)

Bit	Reset	Description
29:24	0x12	RAIL9: Timer Value for rail 9
23:18	0x10	RAIL8: Timer Value for rail 8
17:12	0xe	RAIL7: Timer Value for rail 7
11:6	0xc	RAIL6: Timer Value for rail 6
5:0	0xa	RAIL5: Timer Value for rail 5

10.9.100 APBDEV_PMC_PWRGATE_TIMER_CE_2_0

PWRGATE_TIMER CE_2 removed but kept for backward compatibility.

Offset: 0x18c | Read/Write: R/W | Reset: 0x1c698594 (0bxx0111000110100110000101100101 00)

Bit	Reset	Description
29:24	0x1c	RAIL14: Timer Value for rail 14
23:18	0x1a	RAIL13: Timer Value for rail 13
17:12	0x18	RAIL12: Timer Value for rail 12
11:6	0x16	RAIL11: Timer Value for rail 11
5:0	0x14	RAIL10: Timer Value for rail 10

10.9.101 APBDEV_PMC_PWRGATE_TIMER_CE_3_0

PWRGATE_TIMER CE_3 removed but kept for backward compatibility.

Offset: 0x190 │ Read/Write: R/W │ Reset: 0x2692281e (0bxx100110100100100010100000011110)

Bit	Reset	Description
29:24	0x26	RAIL19: Timer Value for rail 19
23:18	0x24	RAIL18: Timer Value for rail 18
17:12	0x22	RAIL17: Timer Value for rail 17
11:6	0x20	RAIL16: Timer Value for rail 16
5:0	0x1e	RAIL15: Timer Value for rail 15

10.9.102 APBDEV_PMC_PWRGATE_TIMER_CE_4_0

PWRGATE_TIMER CE_4 removed but kept for backward compatibility.

Offset: 0x194 │ Read/Write: R/W │ Reset: 0x30bacaa8 (0bxx110000101110101100101010101000)

Bit	Reset	Description
29:24	0x30	RAIL24: Timer Value for rail 24
23:18	0x2e	RAIL23: Timer Value for rail 23
17:12	0x2c	RAIL22: Timer Value for rail 22
11:6	0x2a	RAIL21: Timer Value for rail 21
5:0	0x28	RAIL20: Timer Value for rail 20

10.9.103 APBDEV_PMC_PWRGATE_TIMER_CE_5_0

PWRGATE_TIMER CE_5 removed but kept for backward compatibility.

Offset: 0x198 │ Read/Write: R/W │ Reset: 0x3ae36d32 (0bxx111010111000110110110100110010)

Bit	Reset	Description
29:24	0x3a	RAIL29: Timer Value for rail 29
23:18	0x38	RAIL28: Timer Value for rail 28
17:12	0x36	RAIL27: Timer Value for rail 27
11:6	0x34	RAIL26: Timer Value for rail 26
5:0	0x32	RAIL25: Timer Value for rail 25

10.9.104 APBDEV_PMC_PWRGATE_TIMER_CE_6_0

PWRGATE_TIMER CE_6 removed but kept for backward compatibility.

Offset: 0x19c │ Read/Write: R/W │ Reset: 0x00000f3b (0bxxxxxxxxxxxxxxxxxxxxxx111100111011)

Bit	Reset	Description
11:6	0x3c	RAIL31: Timer Value for rail 31
5:0	0x3b	RAIL30: Timer Value for rail 30

10.9.105 APBDEV_PMC_PCX_EDPD_CNTRL_0

Defunct

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	EN: when set to 1, sets the EDPD to PCX (PLL toggle) 0 = OFF 1 = ON

10.9.106 APBDEV_PMC_OSC_EDPD_OVER_0

This register can be programmed to keep the oscillator ON during LP0 mode. It cuts the latency time on LP0 wake if the oscillator pad does not have to be restarted.

When the oscillator pad is on, during LP0 mode, DSPARE, duty, strength, and bypass are controlled by the fields below.

The oscillator can be in one in four modes during LP0 - in DPD, not in DPD but not enabled, running, or running only when an external request is active.

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x0000007e (0bxxxxxxxxxx00000000000000001111110)

Bit	Reset	Description
22	0x0	OSC_CTRL_SELECT: select whether the CAR's OSC_CTRL or the PMC's OSC_CTRL_OVER affects the oscillator cell . 0 = CAR 1 = PMC
21:20	0x0	XO_LP0_MODE: control the behavior of the oscillator during LP0 (assuming OSC_CTRL_SELECT is set to PMC during LP0). Put the oscillator in DPD mode during LP0, bypass DPD, but do not enable the oscillator during LP0, force the oscillator to run during LP0, or run the oscillator when requested by an external clock request pin 0 = DPD. 1 = OFF 2 = ON 3 = EXT_REQ
19:12	0x0	OSCFI_SPARE: Crystal oscillator spare register control.
11:7	0x0	XODS: Crystal oscillator duty cycle control.
6:1	0x3f	XOFS: Crystal oscillator drive strength control.
0	0x0	XOBP: Crystal oscillator bypass enable (1 = enable bypass).

10.9.107 APBDEV_PMC_CLK_OUT_CNTRL_0

This register controls 3 clock outputs of the Tegra 4 chip. Each of the clock outputs can be in 1 of 4 modes: not running, running when request is active high, running when request is active low, or always running. The clock output when not running can be tristated, high, or low.

The clock source might be from the CAR unit (not available when in LP0 mode), osc, osc_div2, osc_div4.

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:22	0x0	CLK3_SRC_SEL: 0 = OSC_DIV1 1 = OSC_DIV2 2 = OSC_DIV3 3 = CAR

Bit	Reset	Description
21:20	0x0	CLK3_IDLE_STATE: 0 = LOW 1 = HIGH 2 = TRIS
19	0x0	CLK3_RESERVED: reserved
18	0x0	CLK3_FORCE_EN: force the clock to run regardless of ACCEPT_REQ or INVERT_REQ
17	0x0	CLK3_INVERT_REQ:
16	0x0	CLK3_ACCEPT_REQ:
15:14	0x0	CLK2_SRC_SEL: 0 = OSC_DIV1 1 = OSC_DIV2 2 = OSC_DIV3 3 = CAR
13:12	0x0	CLK2_IDLE_STATE: 0 = LOW 1 = HIGH 2 = TRIS
11	0x0	CLK2_RESERVED: reserved
10	0x0	CLK2_FORCE_EN: force the clock to run regardless of ACCEPT_REQ or INVERT_REQ
9	0x0	CLK2_INVERT_REQ:
8	0x0	CLK2_ACCEPT_REQ:
7:6	0x0	CLK1_SRC_SEL: 0 = OSC_DIV1 1 = OSC_DIV2 2 = OSC_DIV3 3 = CAR
5:4	0x0	CLK1_IDLE_STATE: state of the output line before clock request is active 0 = LOW 1 = HIGH 2 = TRIS
3	0x0	CLK1_RESERVED: reserved
2	0x0	CLK1_FORCE_EN: force the clock to run regardless of ACCEPT_REQ or INVERT_REQ
1	0x0	CLK1_INVERT_REQ:
0	0x0	CLK1_ACCEPT_REQ:

10.9.108 APBDEV_PMC_SATA_PWRGT_0

This register controls the SATA PLLs

PLLE or PADPLL can be disabled/enabled by a software or hardware request.

The SATA PWRGT is reserved for backward compatibility.

Offset: 0x1ac │ Read/Write: R/W │ Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00111111)

Bit	Reset	Description
7	0x0	SW_PLL_EDPD: The SATA PLL is put in the DPD state when this bit is set, independently of power-gating - kept only until drivers are fixed. 0 = OFF 1 = ON
6	0x0	PG_INFO: sm2sata_pg_info
5	0x1	PLLE_IDDQ_OVERRIDE_VALUE: 0: The PLLE is powered up. 1: Software can put the

Bit	Reset	Description
		PLLE in IDDQ mode by setting this bit and PLLE_IDDQ_SWCTL (default) 0 = OFF 1 = ON
4	0x1	PLLE_IDDQ_SWCTL: 0: The PLLE is put in IDDQ mode by hardware (SATA +AFI+CAR) signals. 1: The PLLE is put in IDDQ mode by software -- default 0 = OFF 1 = ON
3	0x1	PADPHY_IDDQ_OVERRIDE_VALUE: 0: The PHY is powered up. 1: Software can put the PHY in IDDQ mode by setting this bit and SATA_PADPHY_IDDQ_SWCT (default) 0 = OFF 1 = ON
2	0x1	PADPHY_IDDQ_SWCTL: 0 The SATA PHY is put in IDDQ mode by hardware (SATA +AFI+CAR) signals. 1 The SATA PHY is put in IDDQ mode by software (default) 0 = OFF 1 = ON
1	0x1	PADPLL_IDDQ_OVERRIDE_VALUE: 0: The pad PLL is powered up 1: Software can put the pad PLL in IDDQ mode by setting this bit and SATA_PADPLL_IDDQ_SWCTL (default) 0 = OFF 1 = ON
0	0x1	PADPLL_IDDQ_SWCTL: 0: The SATA pad PLL is put in IDDQ mode by hardware (SATA +AFI+CAR) signals.1: The SATA pad PLL is put in IDDQ mode by software (default) 0 = OFF 1 = ON

10.9.109 APBDEV_PMC_SENSOR_CTRL_0

For sensor shutdown and control.

The sensor control register defines chip behavior when a sensor request is triggered. It can power-gate down all CPUs if enabled. It can trigger a reset (will cause a CPU power request to go down and power-gates down all CPUs)

IMPORTANT-- to preserve RAM content, any reads/writes to scratch registers will be blocked after a sensor triggered reset.

Software must reset BLOCK_SCRATCH_WRITE to enable writes to scratch registers.

Offset: 0x1b0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	BLOCK_SCRATCH_WRITE: Reset by user, set by the PMC when entering sensor reset 0 = OFF 1 = ON
1	0x0	ENABLE_RST: Enables reset on sensor going up. 0 = OFF 1 = ON
0	0x0	ENABLE_PG: Power gates CPUs on temperature sensor going up. 0 = OFF 1 = ON

10.9.110 APBDEV_PMC_RST_STATUS_0

Simplified Reset and Reset Source

Offset: 0x1b4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2:0	0x0	RST_SOURCE: Source of reset 0 = POR

Bit	Reset	Description
		1 = WATCHDOG 2 = SENSOR 3 = SW_MAIN 4 = LP0

10.9.111 APBDEV_PMC_IO_DPD_REQ_0

Puts I/O rails in or out of DPD mode, even though the chip is not in LP0. Multiple bits are allowed to be set at the same time. No new operation will be triggered until previous one completes. Consecutive operations issued by completion time of first one will be dropped. The register is still updated.

Setting IO_DPD_REQ should be preceded by writing to the DPD_SAMPLE registers (to enable a snapshot of data to be driven) and the SEL_DPD_TIM register. SEL_DPD_TIM is set to a safe value, but it can be lowered if SYS clock is slower (a minimum of 200 ns is required).

DPD Request

Offset: 0x1b8 │ Read/Write: R/W │ Reset: 0x00000000 (0b0000000000000x0000000x0xxxxx0000)

Bit	Reset	Description
31:30	0x0	CODE: Code of operation for all set bits 0 = IDLE 1 = DPD_OFF 2 = DPD_ON
29	0x0	USB_IC: Defunct. Puts usb_ic_pad in/out of deep power down mode 0 = OFF 1 = ON
28	0x0	HDMI: Puts hdmi_pad in/out of Deep Power Down mode 0 = OFF 1 = ON
27	0x0	DDR_DATA: Puts xm0_data0_pad, xm0_data1_pad, xm0_data2_pad, xm0_data3_pad, in/out of Deep Power Down mode 0 = OFF 1 = ON
26	0x0	DISC_ADDR_CMD: Defunct 0 = OFF 1 = ON
25	0x0	DDR_ADDR_CMD: Defunct 0 = OFF 1 = ON
24	0x0	POP_ADDR_CMD: Defunct 0 = OFF 1 = ON
23	0x0	POP_CLK: Defunct 0 = OFF 1 = ON
22	0x0	COMP: puts xm0_comp_pd_pad, xm0_comp_pu_pad in/out of deep power down mode
21	0x0	DISC_VTTGEN: puts mem_addr_vttgen_1 mem_vttgen_35 mem_addr_vttgen_5 mem_vttgen_30 mem_vttgen_34 mem_vttgen_27 mem_vttgen_28 in/out of Deep Power Down mode channel 0 all but clocks 0 = OFF 1 = ON
20	0x0	POP_VTTGEN: Defunct. put pvi_vttgen_1_pad in/out of deep power down mode 0 = OFF 1 = ON

Bit	Reset	Description
19	0x0	HSIC: puts HSIC rail in/out of Deep Power Down mode 0 = OFF 1 = ON
17	0x0	AUDIO: Puts audio rail in/out of Deep Power Down mode 0 = OFF 1 = ON
16	0x0	VI: Defunct. Puts vi rail in/out of Deep Power Down mode 0 = OFF 1 = ON
15	0x0	BB: Puts the BB rail in/out of Deep Power Down mode 0 = OFF 1 = ON
14	0x0	UART: Puts the UART rail in/out of Deep Power Down mode 0 = OFF 1 = ON
13	0x0	NAND: Puts the NAND rail in/out of Deep Power Down mode 0 = OFF 1 = ON
12	0x0	USB_BIAS: Puts usb_bias in/out of Deep Power Down mode 0 = OFF 1 = ON
11	0x0	USB2: Puts USB2 in/out of Deep Power Down mode 0 = OFF 1 = ON
9	0x0	USB0: Puts USB0 in/out of Deep Power Down mode 0 = OFF 1 = ON
3	0x0	MIPI_BIAS: Puts mipi_bias in/out of Deep Power Down mode 0 = OFF 1 = ON
2	0x0	DSI: Puts DSI in/out of Deep Power Down mode 0 = OFF 1 = ON
1	0x0	CSIB: Puts CSIB in/out of Deep Power Down mode 0 = OFF 1 = ON
0	0x0	CSIA: Puts CSIA in/out of Deep Power Down mode 0 = OFF 1 = ON

10.9.112 APBDEV_PMC_IO_DPD_STATUS_0

Reflects the DPD status of each I/O rail.

DPD Status

Offset: 0x1bc │ Read/Write: R/W │ Reset: 0x0000000X (0bxx00000000000x0000000x0xxxxxxxxx)

Bit	R/W	Reset	Description
29	RW	0x0	USB_IC: DEFUNCT. usb_ic_pad in/out of Deep Power Down mode 0 = OFF 1 = ON
28	RW	0x0	HDMI: hdmi_pad in Deep Power Down mode 0 = OFF 1 = ON

Bit	R/W	Reset	Description
27	RW	0x0	DDR_DATA: xm0_data0_pad, xm0_data1_pad, xm0_data2_pad, xm0_data3_pad, in Deep Power Down mode 0 = OFF 1 = ON
26	RW	0x0	DISC_ADDR_CMD: DEFUNCT 0 = OFF 1 = ON
25	RW	0x0	DDR_ADDR_CMD: DEFUNCT 0 = OFF 1 = ON
24	RW	0x0	POP_ADDR_CMD: DEFUNCT 0 = OFF 1 = ON
23	RW	0x0	POP_CLK: DEFUNCT 0 = OFF 1 = ON
22	RW	0x0	COMP: Puts xm0_comp_pd_pad, xm0_comp_pu_pad in Deep Power Down mode
21	RW	0x0	DISC_VTTGEN: Puts mem_addr_vttgen_1 mem_vttgen_35 mem_addr_vttgen_5 mem_vttgen_30 mem_vttgen_34 mem_vttgen_27 mem_vttgen_28 in Deep Power Down mode 0 = OFF 1 = ON
20	RW	0x0	POP_VTTGEN: Puts pvi_vttgen_1_pad in Deep Power Down mode 0 = OFF 1 = ON
19	RW	0x0	HSIC: Puts HSIC rail in Deep Power Down mode 0 = OFF 1 = ON
17	RW	0x0	AUDIO: Puts audio rail in Deep Power Down mode 0 = OFF 1 = ON
16	RW	0x0	VI: DEFUNCT. Puts VI rail in Deep Power Down mode 0 = OFF 1 = ON
15	RW	0x0	BB: BB rail in Deep Power Down mode 0 = OFF 1 = ON
14	RW	0x0	UART: UART rail in Deep Power Down mode 0 = OFF 1 = ON
13	RW	0x0	NAND: NAND rail in Deep Power Down mode 0 = OFF 1 = ON
12	RW	0x0	USB_BIAS: usb_bias in Deep Power Down mode 0 = OFF 1 = ON
11	RW	0x0	USB2: USB2 in Deep Power Down mode 0 = OFF 1 = ON
9	RW	0x0	USB0: USB0 in Deep Power Down mode 0 = OFF 1 = ON

Bit	R/W	Reset	Description
3	RO	X	MIPI_BIAS: mipi_bias in Deep Power Down mode 0 = OFF 1 = ON
2	RO	X	DSI: DSI in Deep Power Down mode 0 = OFF 1 = ON
1	RO	X	CSIB: CSIB in Deep Power Down mode 0 = OFF 1 = ON
0	RO	X	CSIA: CSIA in Deep Power Down mode 0 = OFF 1 = ON

10.9.113 APBDEV_PMC_IO_DPD2_REQ_0

Second set of DPD requests due to additional rails.

Offset: 0x1c0 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:30	0x0	CODE: code of operation for all set bits 0 = IDLE 1 = DPD_OFF 2 = DPD_ON
29:25	0x0	NEW_RAILS: Reserved for future rails
24	0x0	DDR1_DATA: Puts xm1_data0_pad, xm1_data1_pad, xm1_data2_pad, xm1_data3_pad in Deep Power Down mode 0 = OFF 1 = ON
23	0x0	DDR1_CLK: XM1 _MCLK 0 = OFF 1 = ON
22	0x0	DDR1_ADDR2_CMD: MEM1_ADDR2_CLKBUF, XM1_MCAS_N, XM1_MRAS_N, XM1_MWE_N, XM1_MA14, XM1_MA[4-9] 0 = OFF 1 = ON
21	0x0	DDR1_ADDR1_CMD: MEM1_ADDR1_CLKBUF, XM1_MCS*, XM1_MCKE*, XM1_ODT* 0 = OFF 1 = ON
20	0x0	DDR1_ADDR0_CMD: MEM1_ADDR0_CLKBUF, XM1_MBA[0-2], XM1_MA[0-3], XM1_MA[10-13], XM1_RESET 0 = OFF 1 = ON
19	0x0	DDR0_CLK: XM0_MCLK 0 = OFF 1 = ON
18	0x0	DDR0_ADDR2_CMD: MEM0_ADDR2_CLKBUF, XM0_MCAS_N, XM0_MRAS_N, XM0_MWE_N, XM0_MA14, XM0_MA[4-9] 0 = OFF 1 = ON
17	0x0	DDR0_ADDR1_CMD: MEM0_ADDR1_CLKBUF, XM0_MCS*, XM0_MCKE*, XM0_ODT* 0 = OFF 1 = ON

Bit	Reset	Description
16	0x0	DDR0_ADDR0_CMD: MEM0_ADDR0_CLKBUF, XM0_MBA[0-2], XM0_MA[0-3], XM0_MA[10-13], XM0_RESET 0 = OFF 1 = ON
15	0x0	DISC_VTTGEN4: channel 1 clock mem_vttgen_33 0 = OFF 1 = ON
14	0x0	DISC_VTTGEN3: channel 0 clock mem_vttgen_26 0 = OFF 1 = ON
13	0x0	DISC_VTTGEN2: Puts mem_addr_vttgen_3 mem_addr_vttgen_4 mem_addr_vttgen_6 mem_vttgen_36 mem_vttgen_37 mem_vttgen_29 mem_vttgen_31 in deep power down mode (channel 1 all but clock) 0 = OFF 1 = ON
12	0x0	CSIE: Puts CSIE in/out of Deep Power Down mode 0 = OFF 1 = ON
11	0x0	CSID: Puts CSID in/out of Deep Power Down mode 0 = OFF 1 = ON
10	0x0	CSIC: Puts CSIC in/out of Deep Power Down mode 0 = OFF 1 = ON
9	0x0	DSID: Puts DSID in/out of Deep Power Down mode 0 = OFF 1 = ON
8	0x0	DSIC: Puts DSIC in/out of Deep Power Down mode 0 = OFF 1 = ON
7	0x0	DSIB: Puts DSIB in/out of Deep Power Down mode 0 = OFF 1 = ON
6	0x0	HV: Puts HV rail in/out of Deep Power Down mode 0 = OFF 1 = ON
5	0x0	RES_RAIL: 0 = OFF 1 = ON
4	0x0	CAM: Puts the camera in/out of Deep Power Down mode 0 = OFF 1 = ON
3	0x0	SDMMC4: Puts SDMMC4 in/out of Deep Power Down mode 0 = OFF 1 = ON
2	0x0	SDMMC3:puts SDMMC3 in/out of Deep Power Down mode 0 = OFF 1 = ON
1	0x0	SDMMC1:puts SDMMC1 in/out of Deep Power Down mode 0 = OFF 1 = ON
0	0x0	PEX_CNTRL: DEFUNCT. Puts pex_cntrl in/out of Deep Power Down mode – defunct. 0 = OFF 1 = ON

10.9.114 APBDEV_PMC_IO_DPD2_STATUS_0

DPD2 Status

Offset: 0x1c4 | Read/Write: R/W | Reset: 0x000000X0 (0bxx00000000000000000000x0000000)

Bit	R/W	Reset	Description
29:25	RW	0x0	NEW_RAILS
24	RW	0x0	DDR1_DATA: Puts xm1_data0_pad, xm1_data1_pad, xm1_data2_pad, xm1_data3_pad in Deep Power Down mode 0 = OFF 1 = ON
23	RW	0x0	DDR1_CLK: XM1 _MCLK 0 = OFF 1 = ON
22	RW	0x0	DDR1_ADDR2_CMD: MEM1_ADDR2_CLKBUF, XM1_MCAS_N, XM1_MRAS_N, XM1_MWE_N, XM1_MA14, XM1_MA[4-9] 0 = OFF 1 = ON
21	RW	0x0	DDR1_ADDR1_CMD: MEM1_ADDR1_CLKBUF, XM1_MCS*, XM1_MCKE*, XM1_ODT* 0 = OFF 1 = ON
20	RW	0x0	DDR1_ADDR0_CMD: MEM1_ADDR0_CLKBUF, XM1_MBA[0-2], XM1_MA[0-3], XM1_MA[10-13], XM1_RESET 0 = OFF 1 = ON
19	RW	0x0	DDR0_CLK: XM1_MCLK 0 = OFF 1 = ON
18	RW	0x0	DDR0_ADDR2_CMD: MEM0_ADDR2_CLKBUF, XM0_MCAS_N, XM0_MRAS_N, XM0_MWE_N, XM0_MA14, XM0_MA[4-9] 0 = OFF 1 = ON
17	RW	0x0	DDR0_ADDR1_CMD: MEM0_ADDR1_CLKBUF, XM0_MCS*, XM0_MCKE*, XM0_ODT* 0 = OFF 1 = ON
16	RW	0x0	DDR0_ADDR0_CMD: MEM0_ADDR0_CLKBUF, XM0_MBA[0-2], XM0_MA[0-3], XM0_MA[10-13], XM0_RESET 0 = OFF 1 = ON
15	RW	0x0	DISC_VTTGEN4: channel 1 clock mem_vttgen_33 0 = OFF 1 = ON
14	RW	0x0	DISC_VTTGEN3: channel 0 clock mem_vttgen_26 0 = OFF 1 = ON
13	RW	0x0	DISC_VTTGEN2: Puts mem_addr_vttgen_3 mem_addr_vttgen_4 mem_addr_vttgen_6 mem_vttgen_36 mem_vttgen_37 mem_vttgen_29 mem_vttgen_31 in Deep Power Down mode (channel 1 all but clock) 0 = OFF 1 = ON
12	RW	0x0	CSIE:CSIE in/out of Deep Power Down mode 0 = OFF 1 = ON
11	RW	0x0	CSID:CSID in/out of Deep Power Down mode 0 = OFF 1 = ON

Bit	R/W	Reset	Description
10	RW	0x0	CSIC:CSIC in/out of Deep Power Down mode 0 = OFF 1 = ON
9	RW	0x0	DSID:DSID in/out of Deep Power Down mode 0 = OFF 1 = ON
8	RW	0x0	DSIC:DSIC in/out of Deep Power Down mode 0 = OFF 1 = ON
7	RO	X	DSIB: DSI in Deep Power Down mode 0 = OFF 1 = ON
6	RW	0x0	HV:HV rail in/out of Deep Power Down mode 0 = OFF 1 = ON
5	RW	0x0	RES_RAIL: reserved 0 = OFF 1 = ON
4	RW	0x0	CAM: CAM in/out of Deep Power Down mode 0 = OFF 1 = ON
3	RW	0x0	SDMMC4: SDMMC4 in/out of Deep Power Down mode 0 = OFF 1 = ON
2	RW	0x0	SDMMC3: SDMMC3 in/out of Deep Power Down mode 0 = OFF 1 = ON
1	RW	0x0	SDMMC1: SDMMC1 in/out of Deep Power Down mode 0 = OFF 1 = ON
0	RW	0x0	PEX_CNTRL:DEFUNCT 0 = OFF 1 = ON

10.9.115 APBDEV_PMC_SEL_DPD_TIM_0

This timer guarantees proper timing spacing in hardware between the sel_dpd and e_dpd signals issued to pads.

A minimum of 200 ns is required, timer in APB/SYS clock.

Offset: 0x1c8 │ Read/Write: R/W │ Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	SEL_DPD_TIM: timer which separates e_dpd deassertion time from sel_dpd deassertion time in apb_clk units.

10.9.116 APBDEV_PMC_VDDP_SEL_0

Power set for new DDR pads. Safe value is 11.

Offset: 0x1cc │ Read/Write: R/W │ Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11)

Bit	Reset	Description
1:0	0x3	DATA: VDDP sel bits to DDR pads

10.9.117 APBDEV_PMC_DDR_CFG_0

Package Type for CAR/PMC Control

Offset: 0x1d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11	0x0	XM1_MCLK_TRI: overrides to tristate 0 = DISABLE 1 = ENABLE
10	0x0	XM0_MCLK_TRI: overrides to tristate 0 = DISABLE 1 = ENABLE
9	0x0	XM1_TRI: overrides to tristate XM1_MCS* XM1_MCKE* XM1_ODT* 0 = DISABLE 1 = ENABLE
8	0x0	XM0_TRI: overrides to tristate XM0_MCS* XM0_MCKE* XM0_ODT* 0 = DISABLE 1 = ENABLE
7:4	0x0	RESET_SWIZZLE: swizlin reset on any of the pads 0 = XM0_RESET 1 = XM0_MBA0 2 = XM0_MBA1 3 = XM0_MBA2 4 = XM0_MA0 5 = XM0_MA1 6 = XM0_MA2 7 = XM0_MA3 8 = XM0_MA10 9 = XM0_MA11 10 = XM0_MA12 11 = XM0_MA13 12 = NONE
3	0x0	DDR_SPARE
2	0x0	CHAN_SWIZZLE: The EMC uses this to enable pin swizzling across the address pins among channel 0 and channel 1. 0 = DISABLE 1 = ENABLE
1	0x0	IF: 0 = LPDDR2 1 = DDR3
0	0x0	PKG: package type 0 = DISC 1 = POP

10.9.118 APBDEV_PMC_E_NO_VTTGEN_0

Offset: 0x1d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:5	0x0	MASK_XM1: when set, sets E_NO_VTTGEN to 1 on rail during DPD mode only, (turns off DDR3 VTTGEN cell) assignment of the bits. bit 0 mem_addr_vttgen_1_pad_e_no_vttgen bit 1 mem_addr_vttgen_5_pad_e_no_vttgen bit 2 mem_vttgen_30_pad_e_no_vttgen bit 3 mem_vttgen_34_pad_e_no_vttgen bit 4 mem_vttgen_28_pad_e_no_vttgen bit 5 mem_addr_vttgen_3_pad_e_no_vttgen bit 6 mem_addr_vttgen_6_pad_e_no_vttgen bit 7 mem_vttgen_37_pad_e_no_vttgen bit 8 mem_vttgen_36_pad_e_no_vttgen bit 9 mem_vttgen_29_pad_e_no_vttgen
4:0	0x0	MASK: when set,E_NO_VTTGEN is set to 1 on rail during DPD mode only, (turns of DDR3 VTTGEN cell) -xm0

10.9.119 APBDEV_PMC_PLLM_WB0_OVERRIDE_FREQ_0

PLL WB Override Registers to Accelerate Warm Boot Time

Offset: 0x1dc │ Read/Write: R/W │ Reset: 0x0000010c (0bxxxxxxxxxxxxxxxx0000000100001100)

Bit	Reset	Description
15:8	0x1	PLLM_DIVN: PLL feedback divider.
7:0	0xc	PLLM_DIVM: PLL input divider.

10.9.120 APBDEV_PMC_TEST_PWRGATE_0

Force Test Power Gate Override Off/On

Offset: 0x1e0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	RESET_DEBUG: used for debug, assertion of reset will be disabled, if this bit is set 0 = OFF 1 = ON
3	0x0	DPD_ENABLE_DEBUG: used for debug, assertion of dpd_enable will be disabled if this bit is set 0 = OFF 1 = ON
2	0x0	MAIN_CLAMP_DEBUG: used for debug, assertion of main clamp will be disabled if this bit is set 0 = OFF 1 = ON
1:0	0x0	OP: FORCE_ON - force power gated partition to be powered, FORCE_OFF - force power gating partition to be powered down 0 = NONE 1 = FORCE_ON 2 = FORCE_OFF

10.9.121 APBDEV_PMC_PWRGATE_TIMER_MULT_0

The time for each rail set by PWRGATE_TIMER_OFF will be multiplied by MULT for power-gating up/down any power-gated region except CPU. In the CPU power-gated case, MULT_CPU will be used with PWRGATE_TIMER_CE* registers value. All timers are in sys_clk units.

Offset: 0x1e4 │ Read/Write: R/W │ Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0100)

Bit	Reset	Description
3:2	0x1	MULT_CPU: 0 = ONE 1 = TWO 2 = FOUR 3 = EIGHT
1:0	0x0	MULT: 0 = ONE 1 = TWO 2 = FOUR 3 = EIGHT

10.9.122 APBDEV_PMC_DSI_SEL_DPD_0

Register to control sel_dpd for the DSI pad. Allows driving LP0 value on the DSI pad beyond LP0 exit. This enables the DSI to be programmed properly at LP0 exit while the LP0 value is still driven by the brick pad.

Offset: 0x1e8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	SET_DSID: 0 = OFF 1 = ON
2	0x0	SET_DSIC: 0 = OFF 1 = ON
1	0x0	SET_DSIB: 0 = OFF 1 = ON
0	0x0	SET_DSIA: 0 = OFF 1 = ON

10.9.123 APBDEV_PMC_UTMIP_UHSIC_TRIGGERS_0

PMC trigger events for the UTMIP ports as well as the UHSIC port. Sleep walk clearing returns the walk pointer to 00, the first entry in a USB line walk. This is the only way to return a pointer to 0.

Pad configuration captures a set of FLLS parameters prior to resting the port so their value can be retained in deep sleep.

The trigger should only happen on the fields that are set to TRIG (1) when writing. The returned read value should always be all NULL fields.

The FORCE_WALK bits trigger a SLEEP.

Any value read back should be 0.

Triggers for USB Ports

Offset: 0x1ec │ Read/Write: R/W │ Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	UHSIC_CLR_WAKE_ALARM_P0: Clear wake event for UHSIC port 0 0 = NULL 1 = TRIG
14	X	UTMIP_CLR_WAKE_ALARM_P2: Clear wake event for UTMIP port 2 0 = NULL 1 = TRIG
13	X	UTMIP_CLR_WAKE_ALARM_P1: Clear wake event for UTMIP port 1 0 = NULL 1 = TRIG
12	X	UTMIP_CLR_WAKE_ALARM_P0: Clear wake event for UTMIP port 0 0 = NULL 1 = TRIG
11	X	UHSIC_FORCE_WALK_P0: Force pointer walk for UHSIC port 0 0 = NULL 1 = TRIG
10	X	UTMIP_FORCE_WALK_P2: Force pointer walk for UTMIP port 2 0 = NULL 1 = TRIG

Bit	Reset	Description
9	X	UTMIP_FORCE_WALK_P1: Force pointer walk for UTMIP port 1 0 = NULL 1 = TRIG
8	X	UTMIP_FORCE_WALK_P0: Force pointer walk for UTMIP port 0 0 = NULL 1 = TRIG
7	X	UHSIC_RESERVED_P0: Reserved for UHSIC port 0 0 = NULL 1 = TRIG
6	X	UTMIP_CAP_CFG_P2: Capture pad configuration for UTMIP port 2 0 = NULL 1 = TRIG
5	X	UTMIP_CAP_CFG_P1: Capture pad configuration for UTMIP port 1 0 = NULL 1 = TRIG
4	X	UTMIP_CAP_CFG_P0: Capture pad configuration for UTMIP port 0 0 = NULL 1 = TRIG
3	X	UHSIC_CLR_WALK_PTR_P0: Clear sleep walk pointer for UHSIC port 0 0 = NULL 1 = TRIG
2	X	UTMIP_CLR_WALK_PTR_P2: Clear sleep walk pointer for UTMIP port 2 0 = NULL 1 = TRIG
1	X	UTMIP_CLR_WALK_PTR_P1: Clear sleep walk pointer for UTMIP port 1 0 = NULL 1 = TRIG
0	X	UTMIP_CLR_WALK_PTR_P0: Clear sleep walk pointer for UTMIP port 0 0 = NULL 1 = TRIG

10.9.124 APBDEV_PMC_UTMIP_UHSIC_SAVED_STATE_0

Save some critical information about USB port prior to entering DPD in a suspend state.

SPEED:

- HS - Suspend DPD was entered from a high speed mode, restore high speed at DPD exit
- FS - Suspend DPD was entered from a full speed mode, restore full speed at DPD exit
- LS - Suspend DPD was entered from a low speed mode, restore low speed at DPD exit
- RST - Reset Port, Hardware reset or DPD was not entered suspended bus, Reset and Re-enumerate port

SCRATCH -- save other critical information about the port, software choice.

Reset value should read 0x0F0F0F0F, reset should occur on Cold Hardware Reset only

Saved DPD State for all UTMIP and UHSIC Ports

Offset: 0x1f0 │ Read/Write: R/W │ Reset: 0x0f0f0f0f (0b00001111000011110000111100001111)

Bit	Reset	Description
31	0x0	UHSIC_WAKE_EX_P0: Wake up on anything except a Particular Line Value 0 = OFF 1 = ON
30	0x0	UHSIC_IGNORE_MASTER_CFG_P0

Bit	Reset	Description
29:25	0x7	UHSIC_SCRATCH_P0
24	0x1	UHSIC_MODE_P0: UHSIC Speed prior to DPD 0 = HS 1 = RST
23	0x0	UTMIP_WAKE_EX_P2: Wake up on anything except a Particular Line Value 0 = OFF 1 = ON
22	0x0	UTMIP_IGNORE_MASTER_CFG_P2
21:18	0x3	UTMIP_SCRATCH_P2
17:16	0x3	UTMIP_SPEED_P2: UTMIP Speed prior to DPD 0 = HS 1 = FS 2 = LS 3 = RST
15	0x0	UTMIP_WAKE_EX_P1: Wake up on anything except a Particular Line Value 0 = OFF 1 = ON
14	0x0	UTMIP_IGNORE_MASTER_CFG_P1
13:10	0x3	UTMIP_SCRATCH_P1
9:8	0x3	UTMIP_SPEED_P1: UTMIP Speed prior to DPD 0 = HS 1 = FS 2 = LS 3 = RST
7	0x0	UTMIP_WAKE_EX_P0: Wake up on anything except a Particular Line Value 0 = OFF 1 = ON
6	0x0	UTMIP_IGNORE_MASTER_CFG_P0
5:2	0x3	UTMIP_SCRATCH_P0
1:0	0x3	UTMIP_SPEED_P0: UTMIP Speed prior to DPD 0 = HS 1 = FS 2 = LS 3 = RST

10.9.125 APBDEV_PMC_UTMIP_PAD_CFG_0

Set of static configuration values for USB I/O pads under DPD mode. These were configuration values captured at a time prior to entering DPD. These values are read only. Registers must take into account DFT.

I/O Pad Config for Port 0

Offset: 0x1f4 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	X	LSBIAS_SEL_P2: I/O pad LSBIAS_SEL for UTMIP P2
28	X	LO_SPD_P2: I/O pad LO_SPD for UTMIP P2
27:26	X	SPARE_P2: I/O pad SPARE1..0 for UTMIP P2
25:24	X	FS_SLEW_P2: I/O pad FS_SLEW1..0 for UTMIP P2
23:22	X	LS_FSLEW_P2: I/O pad LS_FSLEW1..0 for UTMIP P2
21:20	X	LS_RSLEW_P2: I/O pad LS_RSLEW1..0 for UTMIP P2

Bit	Reset	Description
19	X	LSBIAS_SEL_P1: I/O pad LSBIAS_SEL for UTMIP P1
18	X	LO_SPD_P1: I/O pad LO_SPD for UTMIP P1
17:16	X	SPARE_P1: I/O pad SPARE1..0 for UTMIP P1
15:14	X	FS_SLEW_P1: I/O pad FS_SLEW1..0 for UTMIP P1
13:12	X	LS_FSLEW_P1: I/O pad LS_FSLEW1..0 for UTMIP P1
11:10	X	LS_RSLEW_P1: I/O pad LS_RSLEW1..0 for UTMIP P1
9	X	LSBIAS_SEL_P0: I/O pad LSBIAS_SEL for UTMIP P0
8	X	LO_SPD_P0: I/O pad LO_SPD for UTMIP P0
7:6	X	SPARE_P0: I/O pad SPARE1..0 for UTMIP P0
5:4	X	FS_SLEW_P0: I/O pad FS_SLEW1..0 for UTMIP P0
3:2	X	LS_FSLEW_P0: I/O pad LS_FSLEW1..0 for UTMIP P0
1:0	X	LS_RSLEW_P0: I/O pad LS_RSLEW1..0 for UTMIP P0

10.9.126 APBDEV_PMC_UTMIP_TERM_PAD_CFG_0

Set of termination configuration values for USB I/O pads under DPD mode. These configuration values are programmed, not captured, at a time prior to entering DPD. Capturing them would be complex because it would require costly synchronizers as well as some logic. The range for RCTRL and TCTRL is 0 to 16.

These values need to be converted to a thermal encoding (only one 0 to 1 transition allowed in the word from MSB to LSB). For example:

0 - 0000_0000_0000_0000

1 - 0000_0000_0000_0001

2 - 0000_0000_0000_0011

...

15 - 0111_1111_1111_1111

16 - 1111_1111_1111_1111

I/O Termination Configuration for all UTMIP Ports

Offset: 0x1f8 │ Read/Write: R/W │ Reset: 0x00000108 (0bxxxxxxxxxxxxxxxxxxxxxxxx0100001000)

Bit	Reset	Description
9:5	0x8	TCTRL_VAL: HS termination calibration value, range 0 to 16
4:0	0x8	RCTRL_VAL: 1.5kOhm pull up calibration value, range 0 to 16

10.9.127 APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0

Register that configures the value of the line that could cause a wake-up event.

- ANY -- signifies any line change DM EDGE or DP EDGE for UTMIP.
- NONE -- signifies no wakeup possible.

One can use the two most significant bits (3:2) of the WAKE_VAL to 4x1 select whether:

- 00: Check both bits for value on two LSB (1:0).
- 01: Check bit DM (or strobe for HSIC) against bit 0

- 10: Check bit DP (or DATA for HSIC) against bit 1
- 11: Check for edge on DP against bit 1, edge on DM against bit 0

Sleep Walk Sequence Enables

Offset: 0x1fc │ Read/Write: R/W │ Reset: 0xc0c0c0c0 (0b11000000110000001100000011000000)

Bit	Reset	Description
31:28	0xc	UHSIC_WAKE_VAL_P0: Line Value Wake Up Condition on UHSIC P0 0 = SD00 1 = SD01 2 = SD10 3 = SD11 4 = S0 5 = S1 8 = D0 10 = D1 12 = NONE 13 = SEDGE 14 = DEDGE 15 = ANY
27:25	0x0	UHSIC_RESERVED_P0: Reserved for later use for UHSIC P0
24	0x0	UHSIC_MASTER_ENABLE_P0: Enable use of master pins on UHSIC P0
23:20	0xc	UTMIP_WAKE_VAL_P2: Line Value Wake Up Condition on UTMIP P2 0 = SE0 1 = FSK 2 = FSJ 3 = SE1 4 = DM0 5 = DM1 8 = DP0 10 = DP1 12 = NONE 13 = DMEDGE 14 = DPEDGE 15 = ANY
19	0x0	UTMIP_TCTRL_USE_PMC_P2: Use PMC Saved TCTRL on UTMIP P2
18	0x0	UTMIP_RCTRL_USE_PMC_P2: Use PMC Saved RCTRL on UTMIP P2
17	0x0	UTMIP_FSLS_USE_PMC_P2: Use PMC Saved Pad config on UTMIP P2
16	0x0	UTMIP_MASTER_ENABLE_P2: Enable use of master pins on UTMIP P2
15:12	0xc	UTMIP_WAKE_VAL_P1: Line Value Wake Up Condition on UTMIP P1 0 = SE0 1 = FSK 2 = FSJ 3 = SE1 4 = DM0 5 = DM1 8 = DP0 10 = DP1 12 = NONE 13 = DMEDGE 14 = DPEDGE 15 = ANY
11	0x0	UTMIP_TCTRL_USE_PMC_P1: Use PMC Saved TCTRL on UTMIP P1
10	0x0	UTMIP_RCTRL_USE_PMC_P1: Use PMC Saved RCTRL on UTMIP P1
9	0x0	UTMIP_FSLS_USE_PMC_P1: Use PMC Saved Pad config on UTMIP P1
8	0x0	UTMIP_MASTER_ENABLE_P1: Enable use of master pins on UTMIP P1
7:4	0xc	UTMIP_WAKE_VAL_P0: Line Value Wake Up Condition on UTMIP P0

Bit	Reset	Description
		0 = SE0 1 = FSK 2 = FSJ 3 = SE1 4 = DM0 5 = DM1 8 = DP0 10 = DP1 12 = NONE 13 = DMEDGE 14 = DPEDGE 15 = ANY
3	0x0	UTMIP_TCTRL_USE_PMC_P0: Use PMC Saved TCTRL on UTMIP P0
2	0x0	UTMIP_RCTRL_USE_PMC_P0: Use PMC Saved RCTRL on UTMIP P0
1	0x0	UTMIP_FSLS_USE_PMC_P0: Use PMC Saved Pad config on UTMIP P0
0	0x0	UTMIP_MASTER_ENABLE_P0: Enable use of master pins on UTMIP P0

10.9.128 APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0

This register determines when sleep walking should take effect: upon a specific GPIO event, on any wake event, or on a line value change.

It is also possible to force a sleep walk; see register UTMIP_UHSIC_TRIGGERS to force the event.

Sleep Walk Sequence Enables

Offset: 0x200 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	UHSIC_LINEVAL_WALK_EN_P0: Perform Walk on USB line value wake up for UHSIC P0
30	0x0	UHSIC_WAKE_WALK_EN_P0: Perform Walk on any chip wake up event for UHSIC P0
29	0x0	UHSIC_GPIO_WALK_EN_P0: Perform Walk on associated GPIO event for UHSIC P0
28:24	0x0	UHSIC_DESIGNATED_GPIO_P0: GPIO Number associated with UHSIC P0
23	0x0	UTMIP_LINEVAL_WALK_EN_P2: Perform Walk on USB line value wake up for UTMIP P2
22	0x0	UTMIP_WAKE_WALK_EN_P2: Perform Walk on any chip wake up event for UTMIP P2
21	0x0	UTMIP_GPIO_WALK_EN_P2: Perform Walk on associated GPIO event for UTMIP P2
20:16	0x0	UTMIP_DESIGNATED_GPIO_P2: GPIO Number associated with UTMIP P2
15	0x0	UTMIP_LINEVAL_WALK_EN_P1: Perform Walk on USB line value wake up for UTMIP P1
14	0x0	UTMIP_WAKE_WALK_EN_P1: Perform Walk on any chip wake up event for UTMIP P1
13	0x0	UTMIP_GPIO_WALK_EN_P1: Perform Walk on associated GPIO event for UTMIP P1
12:8	0x0	UTMIP_DESIGNATED_GPIO_P1: GPIO Number associated with UTMIP P1
7	0x0	UTMIP_LINEVAL_WALK_EN_P0: Perform Walk on USB line value wake up for UTMIP P0
6	0x0	UTMIP_WAKE_WALK_EN_P0: Perform Walk on any chip wake up event for UTMIP P0
5	0x0	UTMIP_GPIO_WALK_EN_P0: Perform Walk on associated GPIO event for UTMIP P0
4:0	0x0	UTMIP_DESIGNATED_GPIO_P0: GPIO Number associated with UTMIP P0

10.9.129 APBDEV_PMC_UTMIP_SLEEPWALK_P0_0

These registers hold the sequence of control values to the UTMIP pads on the four consecutive cycles of a walk. The pad pins that are controlled are:

- MASTER_USBOP_RPD
- MASTER_USBON_RPD
- MASTER_USBOP_RPU
- MASTER_USBON_RPU
- MASTER_AP
- MASTER_AN
- MASTER_HIGHZ

For the walk to take effect at the pad, the MASTER_ENABLE pin must be set high in the config register. Otherwise the pad will ignore the values.

If no walk is enabled or forced, then the walk pointer remains stuck on phase A. The walk pointer should use a 2 bit Gray code so that Phase A is 00, Phase B is 01, Phase C is 11 and Phase D is 10. Once Phase D is reached, only a reset of the phase pointer can bring it back to Phase A.

Four phases should be sufficient to handle most wake-up events.

Signaling Sequence for UTMIP Port 0 Wakeup

Offset: 0x204 │ Read/Write: R/W │ Reset: 0x23232363 (0b00100011001000110010001101100011)

Bit	Reset	Description
31	0x0	RESERVED_D: Phase D Unused Bit
30	0x0	HIGHZ_D: Phase D Enable Single Ended Drivers
29	0x1	AN_D: Phase D Drive Single Ended Value on D- line
28	0x0	AP_D: Phase D Drive Single Ended Value on D+ line
27	0x0	USBON_RPU_D: Phase D 1.5kOhm Pull up on D- Line
26	0x0	USBOP_RPU_D: Phase D 1.5kOhm Pull Up on D+ Line
25	0x1	USBON_RPD_D: Phase D 15kOhm Pull Down on D- Line
24	0x1	USBOP_RPD_D: Phase D 15kOhm Pull Down on D+ Line
23	0x0	RESERVED_C: Phase C Unused Bit
22	0x0	HIGHZ_C: Phase C Enable Single Ended Drivers
21	0x1	AN_C: Phase C Drive Single Ended Value on D- line
20	0x0	AP_C: Phase C Drive Single Ended Value on D+ line
19	0x0	USBON_RPU_C: Phase C 1.5kOhm Pull up on D- Line
18	0x0	USBOP_RPU_C: Phase C 1.5kOhm Pull Up on D+ Line
17	0x1	USBON_RPD_C: Phase C 15kOhm Pull Down on D- Line
16	0x1	USBOP_RPD_C: Phase C 15kOhm Pull Down on D+ Line
15	0x0	RESERVED_B: Phase B Unused Bit
14	0x0	HIGHZ_B: Phase B Enable Single Ended Drivers
13	0x1	AN_B: Phase B Drive Single Ended Value on D- line
12	0x0	AP_B: Phase B Drive Single Ended Value on D+ line

Bit	Reset	Description
11	0x0	USBON_RPU_B: Phase B 1.5kOhm Pull up on D- Line
10	0x0	USBOP_RPU_B: Phase B 1.5kOhm Pull Up on D+ Line
9	0x1	USBON_RPD_B: Phase B 15kOhm Pull Down on D- Line
8	0x1	USBOP_RPD_B: Phase B 15kOhm Pull Down on D+ Line
7	0x0	RESERVED_A: Phase A Unused Bit
6	0x1	HIGHZ_A: Phase A Enable Single Ended Drivers
5	0x1	AN_A: Phase A Drive Single Ended Value on D- line
4	0x0	AP_A: Phase A Drive Single Ended Value on D+ line
3	0x0	USBON_RPU_A: Phase A 1.5kOhm Pull up on D- Line
2	0x0	USBOP_RPU_A: Phase A 1.5kOhm Pull Up on D+ Line
1	0x1	USBON_RPD_A: Phase A 15kOhm Pull Down on D- Line
0	0x1	USBOP_RPD_A: Phase A 15kOhm Pull Down on D+ Line

10.9.130 APBDEV_PMC_UTMIP_SLEEPWALK_P1_0

Signaling Sequence for UTMIP Port 1 Wakeup

Offset: 0x208 │ Read/Write: R/W │ Reset: 0x23232363 (0b00100011001000110010001101100011)

Bit	Reset	Description
31	0x0	RESERVED_D: Phase D Unused Bit
30	0x0	HIGHZ_D: Phase D Enable Single Ended Drivers
29	0x1	AN_D: Phase D Drive Single Ended Value on D- line
28	0x0	AP_D: Phase D Drive Single Ended Value on D+ line
27	0x0	USBON_RPU_D: Phase D 1.5kOhm Pull up on D- Line
26	0x0	USBOP_RPU_D: Phase D 1.5kOhm Pull Up on D+ Line
25	0x1	USBON_RPD_D: Phase D 15kOhm Pull Down on D- Line
24	0x1	USBOP_RPD_D: Phase D 15kOhm Pull Down on D+ Line
23	0x0	RESERVED_C: Phase C Unused Bit
22	0x0	HIGHZ_C: Phase C Enable Single Ended Drivers
21	0x1	AN_C: Phase C Drive Single Ended Value on D- line
20	0x0	AP_C: Phase C Drive Single Ended Value on D+ line
19	0x0	USBON_RPU_C: Phase C 1.5kOhm Pull up on D- Line
18	0x0	USBOP_RPU_C: Phase C 1.5kOhm Pull Up on D+ Line
17	0x1	USBON_RPD_C: Phase C 15kOhm Pull Down on D- Line
16	0x1	USBOP_RPD_C: Phase C 15kOhm Pull Down on D+ Line
15	0x0	RESERVED_B: Phase B Unused Bit
14	0x0	HIGHZ_B: Phase B Enable Single Ended Drivers
13	0x1	AN_B: Phase B Drive Single Ended Value on D- line
12	0x0	AP_B: Phase B Drive Single Ended Value on D+ line
11	0x0	USBON_RPU_B: Phase B 1.5kOhm Pull up on D- Line

Bit	Reset	Description
10	0x0	USBOP_RPU_B: Phase B 1.5kOhm Pull Up on D+ Line
9	0x1	USBON_RPD_B: Phase B 15kOhm Pull Down on D- Line
8	0x1	USBOP_RPD_B: Phase B 15kOhm Pull Down on D+ Line
7	0x0	RESERVED_A: Phase A Unused Bit
6	0x1	HIGHZ_A: Phase A Enable Single Ended Drivers
5	0x1	AN_A: Phase A Drive Single Ended Value on D- line
4	0x0	AP_A: Phase A Drive Single Ended Value on D+ line
3	0x0	USBON_RPU_A: Phase A 1.5kOhm Pull up on D- Line
2	0x0	USBOP_RPU_A: Phase A 1.5kOhm Pull Up on D+ Line
1	0x1	USBON_RPD_A: Phase A 15kOhm Pull Down on D- Line
0	0x1	USBOP_RPD_A: Phase A 15kOhm Pull Down on D+ Line

10.9.131 APBDEV_PMC_UTMIP_SLEEPWALK_P2_0

Signaling Sequence for UTMIP Port 2 Wakeup

Offset: 0x20c │ Read/Write: R/W │ Reset: 0x23232363 (0b00100011001000110010001101100011)

Bit	Reset	Description
31	0x0	RESERVED_D: Phase D Unused Bit
30	0x0	HIGHZ_D: Phase D Enable Single Ended Drivers, active low
29	0x1	AN_D: Phase D Drive Single Ended Value on D- line
28	0x0	AP_D: Phase D Drive Single Ended Value on D+ line
27	0x0	USBON_RPU_D: Phase D 1.5kOhm Pull up on D- Line
26	0x0	USBOP_RPU_D: Phase D 1.5kOhm Pull Up on D+ Line
25	0x1	USBON_RPD_D: Phase D 15kOhm Pull Down on D- Line
24	0x1	USBOP_RPD_D: Phase D 15kOhm Pull Down on D+ Line
23	0x0	RESERVED_C: Phase C Unused Bit
22	0x0	HIGHZ_C: Phase C Enable Single Ended Drivers, active low
21	0x1	AN_C: Phase C Drive Single Ended Value on D- line
20	0x0	AP_C: Phase C Drive Single Ended Value on D+ line
19	0x0	USBON_RPU_C: Phase C 1.5kOhm Pull up on D- Line
18	0x0	USBOP_RPU_C: Phase C 1.5kOhm Pull Up on D+ Line
17	0x1	USBON_RPD_C: Phase C 15kOhm Pull Down on D- Line
16	0x1	USBOP_RPD_C: Phase C 15kOhm Pull Down on D+ Line
15	0x0	RESERVED_B: Phase B Unused Bit
14	0x0	HIGHZ_B: Phase B Enable Single Ended Drivers, active low
13	0x1	AN_B: Phase B Drive Single Ended Value on D- line
12	0x0	AP_B: Phase B Drive Single Ended Value on D+ line
11	0x0	USBON_RPU_B: Phase B 1.5kOhm Pull up on D- Line
10	0x0	USBOP_RPU_B: Phase B 1.5kOhm Pull Up on D+ Line

Bit	Reset	Description
9	0x1	USBON_RPD_B: Phase B 15kOhm Pull Down on D- Line
8	0x1	USBOP_RPD_B: Phase B 15kOhm Pull Down on D+ Line
7	0x0	RESERVED_A: Phase A Unused Bit
6	0x1	HIGHZ_A: Phase A Enable Single Ended Drivers, active low
5	0x1	AN_A: Phase A Drive Single Ended Value on D- line
4	0x0	AP_A: Phase A Drive Single Ended Value on D+ line
3	0x0	USBON_RPU_A: Phase A 1.5kOhm Pull up on D- Line
2	0x0	USBOP_RPU_A: Phase A 1.5kOhm Pull Up on D+ Line
1	0x1	USBON_RPD_A: Phase A 15kOhm Pull Down on D- Line
0	0x1	USBOP_RPD_A: Phase A 15kOhm Pull Down on D+ Line

10.9.132 APBDEV_PMC_UHSIC_SLEEPWALK_P0_0

These registers hold the sequence of control values to the UTMIP pads on the four consecutive cycles of a walk. The pad pins that are controlled are:

- STROBE_RPD
- DATA_RPD
- STROBE_RPU
- DATA_RPU

At this time there are no STROBE, DATA ports that should be driven by the host. So these outputs from the module should be left unconnected until a time that departure may be implemented.

Signaling Sequence for UHSIC Port 0 Wakeup

Offset: 0x210 │ Read/Write: R/W │ Reset: 0x06060606 (0b00000110000001100000011000000110)

Bit	Reset	Description
31:28	0x0	RESERVED_D: Phase D Unused Bit
27	0x0	UHSIC_DATA_RPU_D: Phase D Pull up on DATA Line
26	0x1	UHSIC_STROBE_RPU_D: Phase D Pull Up on STROBE Line
25	0x1	UHSIC_DATA_RPD_D: Phase D Pull Down on DATA Line
24	0x0	UHSIC_STROBE_RPD_D: Phase D Pull Down on STROBE Line
23:20	0x0	RESERVED_C: Phase C Unused Bit
19	0x0	UHSIC_DATA_RPU_C: Phase C Pull up on DATA Line
18	0x1	UHSIC_STROBE_RPU_C: Phase C Pull Up on STROBE Line
17	0x1	UHSIC_DATA_RPD_C: Phase C Pull Down on DATA Line
16	0x0	UHSIC_STROBE_RPD_C: Phase C Pull Down on STROBE Line
15:12	0x0	RESERVED_B: Phase B Unused Bit
11	0x0	UHSIC_DATA_RPU_B: Phase B Pull up on DATA Line
10	0x1	UHSIC_STROBE_RPU_B: Phase B Pull Up on STROBE Line
9	0x1	UHSIC_DATA_RPD_B: Phase B Pull Down on DATA Line
8	0x0	UHSIC_STROBE_RPD_B: Phase B Pull Down on STROBE Line

Bit	Reset	Description
7:4	0x0	RESERVED_A: Phase A Unused Bit
3	0x0	UHSIC_DATA_RPU_A: Phase A Pull up on DATA Line
2	0x1	UHSIC_STROBE_RPU_A: Phase A Pull Up on STROBE Line
1	0x1	UHSIC_DATA_RPD_A: Phase A Pull Down on DATA Line
0	0x0	UHSIC_STROBE_RPD_A: Phase A Pull Down on STROBE Line

10.9.133 APBDEV_PMC_UTMIP_UHSIC_STATUS_0

Read-only register that provides current walk pointer information as well as line value.

Status of UTMIP UHSIC Wake-up Circuitry

Offset: 0x214 │ Read/Write: RO │ Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19	X	UHSIC_WAKE_ALARM_P0: A wake event occurred on UHSIC port 0
18	X	UTMIP_WAKE_ALARM_P2: A wake event occurred on UTMIP port 2
17	X	UTMIP_WAKE_ALARM_P1: A wake event occurred on UTMIP port 1
16	X	UTMIP_WAKE_ALARM_P0: A wake event occurred on UTMIP port 0
15	X	DATA_VAL_P0: Value of DATA line detector for UHSIC port 0
14	X	STROBE_VAL_P0: Value of STROBE line detector for UHSIC port 0
13	X	USBON_VAL_P2: Value of D- line detector for UTMIP port 2
12	X	USBOP_VAL_P2: Value of D+ line detector for UTMIP port 2
11	X	USBON_VAL_P1: Value of D- line detector for UTMIP port 1
10	X	USBOP_VAL_P1: Value of D+ line detector for UTMIP port 1
9	X	USBON_VAL_P0: Value of D- line detector for UTMIP port 0
8	X	USBOP_VAL_P0: Value of D+ line detector for UTMIP port 0
7:6	X	UHSIC_WALK_PTR_P0: Walk pointer for UHSIC port 0
5:4	X	UTMIP_WALK_PTR_P2: Walk pointer for UTMIP port 2
3:2	X	UTMIP_WALK_PTR_P1: Walk pointer for UTMIP port 1
1:0	X	UTMIP_WALK_PTR_P0: Walk pointer for UMTIP port 0

10.9.134 APBDEV_PMC_UTMIP_UHSIC_FAKE_0

Instead of using the pad value for USBOP_VAL, USBON_VAL, STROBE_VAL, DATA_VAL, VBUS_WAKEUP, ID from the UTMIP and HSIC pads, force a 2x1 mux at the input of the PMC pad macro when fake values are enabled. This could the useful feature of putting a line at rest when needed. It also has important debug merits. This mux should be placed in front of the NP synchronizer for these signals (saved power on the synchronizer). This will have to be waived in our sync checks.

Fake the Line Value for the PMC Pad Macro

Offset: 0x218 │ Read/Write: R/W │ Reset: 0x01111111 (0bxxxx00010001000100010001000100010001)

Bit	Reset	Description
27	0x0	UTMIP_ID_FAKE_EN_P2: Enable the fake ID value for UTMIP P2

Bit	Reset	Description
26	0x0	UTMIP_ID_FAKE_VAL_P2: Fake ID value for UTMIP P2
25	0x0	UTMIP_VBUS_FAKE_EN_P2: Enable the fake VBUS WAKEUP value for UTMIP P2
24	0x1	UTMIP_VBUS_FAKE_VAL_P2: Fake VBUS WAKEUP value for UTMIP P2
23	0x0	UTMIP_ID_FAKE_EN_P1: Enable the fake ID value for UTMIP P1
22	0x0	UTMIP_ID_FAKE_VAL_P1: Fake ID value for UTMIP P1
21	0x0	UTMIP_VBUS_FAKE_EN_P1: Enable the fake VBUS WAKEUP value for UTMIP P1
20	0x1	UTMIP_VBUS_FAKE_VAL_P1: Fake VBUS WAKEUP value for UTMIP P1
19	0x0	UTMIP_ID_FAKE_EN_P0: Enable the fake ID value for UTMIP P0
18	0x0	UTMIP_ID_FAKE_VAL_P0: Fake ID value for UTMIP P0
17	0x0	UTMIP_VBUS_FAKE_EN_P0: Enable the fake VBUS WAKEUP value for UTMIP P0
16	0x1	UTMIP_VBUS_FAKE_VAL_P0: Fake VBUS WAKEUP value for UTMIP P0
15	0x0	UHSIC_FAKE_DATA_EN_P0: Enable the fake line value for DATA for the PMC pad macro for UHSIC P0
14	0x0	UHSIC_FAKE_STROBE_EN_P0: Enable the fake line value for STROBE for the PMC pad macro for UHSIC P0
13	0x0	UHSIC_FAKE_DATA_VAL_P0: Fake line value for DATA for the PMC pad macro for UHSIC P0
12	0x1	UHSIC_FAKE_STROBE_VAL_P0: Fake line value for STROBE for the PMC pad macro for UHSIC P0
11	0x0	UTMIP_FAKE_USBON_EN_P2: Enable the fake line value for D- for the PMC pad macro for UTMIP P2
10	0x0	UTMIP_FAKE_USBOP_EN_P2: Enable the fake line value for D+ for the PMC pad macro for UTMIP P2
9	0x0	UTMIP_FAKE_USBON_VAL_P2: Fake line value for D- for the PMC pad macro for UTMIP P2
8	0x1	UTMIP_FAKE_USBOP_VAL_P2: Fake line value for D+ for the PMC pad macro for UTMIP P2
7	0x0	UTMIP_FAKE_USBON_EN_P1: Enable the fake line value for D- for the PMC pad macro for UTMIP P1
6	0x0	UTMIP_FAKE_USBOP_EN_P1: Enable the fake line value for D+ for the PMC pad macro for UTMIP P1
5	0x0	UTMIP_FAKE_USBON_VAL_P1: Fake line value for D- for the PMC pad macro for UTMIP P1
4	0x1	UTMIP_FAKE_USBOP_VAL_P1: Fake line value for D+ for the PMC pad macro for UTMIP P1
3	0x0	UTMIP_FAKE_USBON_EN_P0: Enable the fake line value for D- for the PMC pad macro for UTMIP P0
2	0x0	UTMIP_FAKE_USBOP_EN_P0: Enable the fake line value for D+ for the PMC pad macro for UTMIP P0
1	0x0	UTMIP_FAKE_USBON_VAL_P0: Fake line value for D- for the PMC pad macro for UTMIP P0
0	0x1	UTMIP_FAKE_USBOP_VAL_P0: Fake line value for D+ for the PMC pad macro for UTMIP P0

10.9.135 APBDEV_PMC_BONDOUT_MIRROR3_0

Secure Scratch Register

Offset: 0x21c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR3

10.9.136 APBDEV_PMC_BONDOUT_MIRROR4_0

Secure Scratch Register

Offset: 0x220 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR4

10.9.137 APBDEV_PMC_SECURE_SCRATCH6_0

Secure Scratch Register

Offset: 0x224 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH6

10.9.138 APBDEV_PMC_SECURE_SCRATCH7_0

Secure Scratch Register

Offset: 0x228 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH7

10.9.139 APBDEV_PMC_SCRATCH43_0

Scratch Register

Offset: 0x22c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH43: General-purpose register storage. Currently used for power-gating progress status in RTAPI.

10.9.140 APBDEV_PMC_SCRATCH44_0

Scratch Register

Offset: 0x230 | Read/Write: R/W V Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH44: General-purpose register storage. Currently used for storing ARM SP in power-gating by RTAPI.

10.9.141 APBDEV_PMC_SCRATCH45_0

Scratch Register

Offset: 0x234 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH45: General-purpose register storage

10.9.142 APBDEV_PMC_SCRATCH46_0

Scratch Register

Offset: 0x238 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH46: General-purpose register storage

10.9.143 APBDEV_PMC_SCRATCH47_0

Scratch Register

Offset: 0x23c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH47: General-purpose register storage

10.9.144 APBDEV_PMC_SCRATCH48_0

Scratch Register

Offset: 0x240 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH48: General-purpose register storage

10.9.145 APBDEV_PMC_SCRATCH49_0

Scratch Register

Offset: 0x244 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH49: General-purpose register storage

10.9.146 APBDEV_PMC_SCRATCH50_0

Scratch Register

Offset: 0x248 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH50: General-purpose register storage

10.9.147 APBDEV_PMC_SCRATCH51_0

Scratch Register

Offset: 0x24c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH51: General-purpose register storage

10.9.148 APBDEV_PMC_SCRATCH52_0

Scratch Register

Offset: 0x250 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH52: General-purpose register storage

10.9.149 APBDEV_PMC_SCRATCH53_0

Scratch Register

Offset: 0x254 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH53: General-purpose register storage

10.9.150 APBDEV_PMC_SCRATCH54_0

Scratch Register

Offset: 0x258 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH54: General-purpose register storage

10.9.151 APBDEV_PMC_SCRATCH55_0

Scratch Register

Offset: 0x25c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH55: General-purpose register storage

10.9.152 APBDEV_PMC_SCRATCH0_ECO_0

Scratch Register

Offset: 0x260 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CAR_USE_NEW_LP0_EXIT_SEQ:Reserved.
30:0	0x0	ECO0: Reserved.

10.9.153 APBDEV_PMC_SCRATCH1_ECO_0

Scratch Register

Offset: 0x264 | Read/Write: R/W | Reset: 0x0000003f (0b00000000000000000000000000111111)

Bit	Reset	Description
31:6	0x0	ECO1: Reserved.
5	0x1	MEM1_ADDR2_CLK_SEL_DPD: sets sel_dpd to clock buffer.
4	0x1	MEM1_ADDR1_CLK_SEL_DPD: sets sel_dpd to clock buffer.
3	0x1	MEM1_ADDR0_CLK_SEL_DPD: sets sel_dpd to clock buffer.
2	0x1	MEM0_ADDR2_CLK_SEL_DPD: sets sel_dpd to clock buffer.
1	0x1	MEM0_ADDR1_CLK_SEL_DPD: sets sel_dpd to clock buffer.
0	0x1	MEM0_ADDR0_CLK_SEL_DPD: sets sel_dpd to clock buffer.

10.9.154 APBDEV_PMC_SCRATCH2_ECO_0

Scratch Register

Offset: 0x268 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	ECO2: Reserved.

10.9.155 APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0

Offset: 0x26c | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3	0x1	UHSIC_LINE_WAKEUP_EN_P0: enables latching line wakeup event on UHSIC p0
2	0x1	UTMIP_LINE_WAKEUP_EN_P2: enables latching line wakeup event on UTMIP p2
1	0x1	UTMIP_LINE_WAKEUP_EN_P1: enables latching line wakeup event on UTMIP p1
0	0x1	UTMIP_LINE_WAKEUP_EN_P0: enables latching line wakeup event on UTMIP p0

10.9.156 APBDEV_PMC_UTMIP_BIAS_MASTER_CNTRL_0

Offset: 0x270 │ Read/Write: R/W │ Reset: 0x0000000d (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1101)

Bit	Reset	Description
3	0x1	UTMIP_BIAS_MASTER_AWAKE_AND: When not PROGRAMMABLE and E_DPD=0, AND the MASTER_ENABLE of all IO ports for bias cell. If not AND or OR, force to 0.
2	0x1	UTMIP_BIAS_MASTER_AWAKE_OR: When not PROGRAMMABLE and E_DPD=0, OR the MASTER_ENABLE of all IO ports for bias cell
1	0x0	UTMIP_BIAS_MASTER_PROG_VAL: When PROGRAMMABLE, this is the value to program too
0	0x1	UTMIP_BIAS_MASTER_PROG_CTRL: MASTER_ENABLE pin uses a programmable value on BIAS pad, at all times

10.9.157 APBDEV_PMC_UTMIP_MASTER_CONFIG_0

MASTER_CONFIG selects the MASTER function's mode of operation, in this case:

- 0 for 500 µA usage
- 1 for smaller current usage when driving.

Offset: 0x274 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	UHSIC_PWR_P0: enables UHSIC p0 low power mode
2	0x0	UTMIP_PWR_P2: enables UTMIP p2 low power mode
1	0x0	UTMIP_PWR_P1: enables UTMIP p1 low power mode
0	0x0	UTMIP_PWR_P0: enables UTMIP p0 low power mode

10.9.158 APBDEV_PMC_TD_PWRGATE_INTER_PART_TIMER_0

TD Power-Gating Timing Between Partition Power-Gating

Offset: 0x278 │ Read/Write: R/W │ Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3:0	0xf	DATA: timing between consecutive partitions

10.9.159 APBDEV_PMC_UTMIP_UHSIC2_TRIGGERS_0

PMC trigger events for the UTMIP ports as well as the UHSIC port. Sleep walk clearing returns the walk pointer to 00, the first entry in a USB line walk. This is the only way to return a pointer to 0.

Pad configuration capture a set of FLLS parameters prior to resting the port so their value can be retained in deep sleep.

The trigger should only happen on the fields that are set to TRIG (1) when writing. Returned read value should always be all NULL fields. The FORCE_WALK bits trigger a SLEEP

This does not need to be a real register, so it should not consume much area. Any value read back should be 0.

Triggers for USB Ports

Offset: 0x27c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3	X	UHSIC_CLR_WAKE_ALARM_P1: Clear wake event for UHSIC port 0 0 = NULL 1 = TRIG
2	X	UHSIC_FORCE_WALK_P1: Force pointer walk for UHSIC port 0 0 = NULL 1 = TRIG
1	X	UHSIC_RESERVED_P1: Reserved for UHSIC port 0 0 = NULL 1 = TRIG
0	X	UHSIC_CLR_WALK_PTR_P1: Clear sleep walk pointer for UHSIC port 1 0 = NULL 1 = TRIG

10.9.160 APBDEV_PMC_UTMIP_UHSIC2_SAVED_STATE_0

Save some critical information about USB port prior to entering DPD in a suspend state.

SPEED

- HS: Suspend DPD was entered from a high speed mode, restore high speed at DPD exit
- FS: Suspend DPD was entered from a full speed mode, restore full speed at DPD exit
- LS: Suspend DPD was entered from a low speed mode, restore low speed at DPD exit
- RST: Reset Port, Hardware reset or DPD was not entered suspended bus, Reset and Re-enumerate port

SCRATCH -- Save other critical information about the port, software choice. Reset value should read 0x0F0F0F0F, reset should occur on Cold Hardware Reset only

Saved DPD State for all UTMIP and UHSIC Ports

Offset: 0x280 │ Read/Write: R/W │ Reset: 0x00000007 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000111)

Bit	Reset	Description
7	0x0	UHSIC_WAKE_EX_P1: Wake up on anything except a particular line value 0 = OFF 1 = ON
6	0x0	UHSIC_IGNORE_MASTER_CFG_P1
5:1	0x3	UHSIC_SCRATCH_P1
0	0x1	UHSIC_MODE_P1: UHSIC Speed prior to DPD 0 = HS 1 = RST

10.9.161 APBDEV_PMC_UTMIP_UHSIC2_SLEEP_CFG_0

This register configures the value of the line that could cause a wake-up event.

- ANY: signifies any line change DM EDGE or DP EDGE for UTMIP.
- NONE: signifies no wake-up possible.

One can use the two most significant bits (3:2) of the WAKE_VAL to 4x1 select whether:

- 00: Check both bits for value on two LSB (1:0).
- 01: Check bit DM (or strobe for HSIC) against bit 0
- 10: Check bit DP (or DATA for HSIC) against bit 1
- 11: Check for edge on DP against bit 1, edge on DM against bit 0

Sleep Walk Sequence Enables

Offset: 0x284 │ Read/Write: R/W │ Reset: 0x000000c0 (0bxxxxxxxxxxxxxxxxxxxxxxxx11000000)

Bit	Reset	Description
7:4	0xc	UHSIC_WAKE_VAL_P1: Line Value Wake Up Condition on UHSIC P1 0 = SD00 1 = SD01 2 = SD10 3 = SD11 4 = S0 5 = S1 8 = D0 10 = D1 12 = NONE 13 = SEDGE 14 = DEDGE 15 = ANY
3:1	0x0	UHSIC_RESERVED_P1: Reserved for later use for UHSIC P1
0	0x0	UHSIC_MASTER_ENABLE_P1: Enable use of master pins on UHSIC P1

10.9.162 APBDEV_PMC_UTMIP_UHSIC2_SLEEPWALK_CFG_0

This register determines when sleep walking should take effect: upon a specific GPIO event, on any wake event, or on a line value change.

Note that it is also possible to force a sleep walk: see register UTMIP_UHSIC_TRIGGERS to force the event.

Sleep Walk Sequence Enables

Offset: 0x288 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	UHSIC_LINEVAL_WALK_EN_P1: Perform walk on USB line value wake up for UHSIC P1
6	0x0	UHSIC_WAKE_WALK_EN_P1: Perform walk on any chip wake up event for UHSIC P1
5	0x0	UHSIC_GPIO_WALK_EN_P1: Perform walk on associated GPIO event for UHSIC P1
4:0	0x0	UHSIC_DESIGNATED_GPIO_P1: GPIO Number associated with UHSIC P1

10.9.163 APBDEV_PMC_UHSIC2_SLEEPWALK_P1_0

This register holds the sequence of control values to the UTMIP pads on the four consecutive cycles of a walk. The pad pins that are controlled are:

- STROBE_RPD
- DATA_RPD
- STROBE_RPU
- DATA_RPU

At this time there are no STROBE, DATA ports that should be driven by the host. So these outputs from the module should be left dangling (unconnected) until a time that feature may be implemented.

Signaling Sequence for UHSIC Port 0 Wakeup

Offset: 0x28c │ Read/Write: R/W │ Reset: 0x06060606 (0b00000110000001100000011000000110)

Bit	Reset	Description
31:28	0x0	RESERVED_D: Phase D Unused Bit
27	0x0	UHSIC_DATA_RPU_D: Phase D Pull up on DATA Line
26	0x1	UHSIC_STROBE_RPU_D: Phase D Pull Up on STROBE Line
25	0x1	UHSIC_DATA_RPD_D: Phase D Pull Down on DATA Line
24	0x0	UHSIC_STROBE_RPD_D: Phase D Pull Down on STROBE Line
23:20	0x0	RESERVED_C: Phase C Unused Bit
19	0x0	UHSIC_DATA_RPU_C: Phase C Pull up on DATA Line
18	0x1	UHSIC_STROBE_RPU_C: Phase C Pull Up on STROBE Line
17	0x1	UHSIC_DATA_RPD_C: Phase C Pull Down on DATA Line
16	0x0	UHSIC_STROBE_RPD_C: Phase C Pull Down on STROBE Line
15:12	0x0	RESERVED_B: Phase B Unused Bit
11	0x0	UHSIC_DATA_RPU_B: Phase B Pull up on DATA Line
10	0x1	UHSIC_STROBE_RPU_B: Phase B Pull Up on STROBE Line
9	0x1	UHSIC_DATA_RPD_B: Phase B Pull Down on DATA Line
8	0x0	UHSIC_STROBE_RPD_B: Phase B Pull Down on STROBE Line
7:4	0x0	RESERVED_A: Phase A Unused Bit
3	0x0	UHSIC_DATA_RPU_A: Phase A Pull up on DATA Line
2	0x1	UHSIC_STROBE_RPU_A: Phase A Pull Up on STROBE Line
1	0x1	UHSIC_DATA_RPD_A: Phase A Pull Down on DATA Line
0	0x0	UHSIC_STROBE_RPD_A: Phase A Pull Down on STROBE Line

10.9.164 APBDEV_PMC_UTMIP_UHSIC2_STATUS_0

Read-only register that provides current walk pointer information as well as the line value.

Status of UTMIP UHSIC Wakeup Circuitry

Offset: 0x290 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
4	X	UHSIC_WAKE_ALARM_P1: A wake event occurred on UHSIC port 1
3	X	DATA_VAL_P1: Value of DATA line detector for UHSIC port 1
2	X	STROBE_VAL_P1: Value of STROBE line detector for UHSIC port 1
1:0	X	UHSIC_WALK_PTR_P1: Walk pointer for UHSIC port 0

10.9.165 APBDEV_PMC_UTMIP_UHSIC2_FAKE_0

Instead of using the pad value for USBOP_VAL, USBON_VAL, STROBE_VAL, DATA_VAL, VBUS_WAKEUP, ID from the UTMIP and HSIC pads, force a 2x1 mux at the input of the PMC pad macro when fake values are enabled. This could the useful feature of putting a line at rest when needed. It also has important debug merits. This mux should be placed in front of the NP synchronizer for these signals (saved power on the synchronizer). This will have to be waived in our sync checks.

Fake the Line Value for the PMC Pad Macro

Offset: 0x294 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001)

Bit	Reset	Description
3	0x0	UHSIC_FAKE_DATA_EN_P1: Enable the fake line value for DATA for the PMC pad macro for UHSIC P0
2	0x0	UHSIC_FAKE_STROBE_EN_P1: Enable the fake line value for STROBE for the PMC pad macro for UHSIC P0
1	0x0	UHSIC_FAKE_DATA_VAL_P1: Fake line value for DATA for the PMC pad macro for UHSIC P0
0	0x1	UHSIC_FAKE_STROBE_VAL_P1: Fake line value for STROBE for the PMC pad macro for UHSIC P0

10.9.166 APBDEV_PMC_UTMIP_UHSIC2_LINE_WAKEUP_0

Offset: 0x298 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	UHSIC_LINE_WAKEUP_EN_P1: enables latching line wake-up event on UHSIC P1

10.9.167 APBDEV_PMC_UTMIP_MASTER2_CONFIG_0

MASTER_CONFIG selects the MASTER functions mode of operation, in this case 0 for 500uA usage, 1 for smaller current usage when driving.

Offset: 0x29c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	UHSIC_PWR_P1: enables UHSIC P1 low power mode

10.9.168 APBDEV_PMC_UTMIP_UHSIC_RPD_CFG_0

Offset: 0x2a0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	WEAKPD_ANYTIME_P2
7	0x0	DP_WEAKPD_CFG_P2
6	0x0	DM_WEAKPD_CFG_P2
5	0x0	WEAKPD_ANYTIME_P1
4	0x0	DP_WEAKPD_CFG_P1
3	0x0	DM_WEAKPD_CFG_P1
2	0x0	WEAKPD_ANYTIME_P0
1	0x0	DP_WEAKPD_CFG_P0

Bit	Reset	Description
0	0x0	DM_WEAKPD_CFG_P0

10.9.169 APBDEV_PMC_PG_MASK_CE0_0

Offset: 0x2a4 | Read/Write: R/W | Reset: 0x000000ff (0bxxxxxxxxxxxxxxxxxxxxxxxxx11111111)

Bit	Reset	Description
7:0	0xff	MASK: Mask CE1 rail

10.9.170 APBDEV_PMC_PG_MASK_3_0

Offset: 0x2a8 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:24	0xff	DISB: Mask DISB rail
23:16	0xff	DIS: Mask DIS rail
15:8	0xff	C1NC: Mask C1NC rail
7:0	0xff	C0NC: Mask C0NC rail

10.9.171 APBDEV_PMC_PG_MASK_4_0

Offset: 0x2ac | Read/Write: R/W | Reset: 0x00ffffff (0bxxxxxxxx111111111111111111111111)

Bit	Reset	Description
23:16	0xff	XUSBC: Mask XUSBC rail
15:8	0xff	XUSBB: Mask XUSBB rail
7:0	0xff	XUSBA: Mask XUSBA rail

10.9.172 APBDEV_PMC_PLLM_WB0_OVERRIDE2_0

Offset: 0x2b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000000000000000000000)

Bit	Reset	Description
27	0x0	DIV2: New divide by 2
26	0x0	KVCO: KVCO/VCO gain
25:24	0x0	KCP: Charge pump control
23:0	0x0	SETUP: Setup

10.9.173 APBDEV_PMC_TSC_MULT_0

Offset: 0x2b4 | Read/Write: R/W | Reset: 0x000016e0 (0bxxxxxxxxxxxxxxxx0001011011100000)

Bit	R/W	Reset	Description
16	RO	X	FREQ_STS: Clock frequency being used for counter. 0 = Fast (i.e., oscillator frequency). 1 = Slow (i.e., 32kHz). When the PMC_DPD_ENABLE[TSC_MULT_EN] bit is set to '1', the PMC waits for first rising edge on the slow clock before switching the counter to the slow clock. This status is also set at the first rising edge of the slow clock (to indicate the change of clock for counter).

Bit	R/W	Reset	Description
15:0	RW	0x16e0	MULT_VAL: TSC multiply value (default based on 12 MHz oscillator).Value is (osc-freq * 16 / 32.768 kHz) when the oscillator is disabled and the TSC runs at the 32 kHz clock. For example, for a 12 MHz oscillator, VALUE = 5856.

10.9.174 APBDEV_PMC_CPU_VSENSE_OVERRIDE_0

Offset: 0x2b8 │ Read/Write: R/W │ Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5	0x0	VDD: VDD sensing override.
4	0x1	C0NC: C0NC VVDD partition sensing override.
3	0x1	CE3:CE3 VVDD partition sensing override.
2	0x1	CE2:CE2 VVDD partition sensing override.
1	0x1	CE1:CE1 VVDD partition sensing override.
0	0x1	CE0:CE0 VVDD partition sensing override.

10.9.175 APBDEV_PMC_GLB_AMAP_CFG_0

This register configures some of the address apertures (in CCPLEX-AXD) to be MMIO or DRAM. Each bit is used to configurable one of the address aperture. For each bit 0=>MMIO and 1=>DRAM.

By default, all configurable apertures are MMIO address space. But they can be configured (at boot time) to be DRAM by programming this register. This register can be write-disabled (by the PMC_SEC_DISABLE register). The bits of this register are used as follows.

GLB_AMAP_CFG

Offset: 0x2bc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
17	0x0	IROM_HIVEC aperture: 0 = MMIO 1 = DRAM
16	0x0	AHB_A2_RSVD aperture: 0 = MMIO 1 = DRAM
15	0x0	AHB_A1_RSVD aperture: 0 = MMIO 1 = DRAM
14	0x0	AHB_A1 aperture: 0 = MMIO 1 = DRAM
13	0x0	APB_RSVD aperture: 0 = MMIO 1 = DRAM
12	0x0	EXTIO_RSVD aperture: 0 = MMIO 1 = DRAM
11	0x0	PPSB_RSVD aperture: 0 = MMIO 1 = DRAM

Bit	Reset	Description
10	0x0	GART_GPU aperture: 0 = MMIO 1 = DRAM
9	0x0	GFX_HOST_RSVD aperture: 0 = MMIO 1 = DRAM
8	0x0	VERIF_RSVD aperture: 0 = MMIO 1 = DRAM
7	0x0	NOR_A3 aperture: 0 = MMIO 1 = DRAM
6	0x0	NOR_A2 aperture: 0 = MMIO 1 = DRAM
5	0x0	NOR_A1 aperture: 0 = MMIO 1 = DRAM
4	0x0	IRAM_RSVD aperture: 0 = MMIO 1 = DRAM
3	0x0	PCIE_A3 aperture: 0 = MMIO 1 = DRAM
2	0x0	PCIE_A2 aperture: 0 = MMIO 1 = DRAM
1	0x0	PCIE_A1 aperture: 0 = MMIO 1 = DRAM
0	0x0	IROM_LOVEC aperture: 0 = MMIO 1 = DRAM

10.9.176 APBDEV_PMC_STICKY_BITS_0

Offset: 0x2c0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6	0x0	JTAG_STS: Secure Sticky one bit to propagate JTAG enable/disable across LP0.This bit is set to '1' only by a secure write of '1'. All other writes are ignored. Once it is set to '1', it remains '1' and only resets by system reset (same resets which resets main PMC). 0 = ENABLE 1 = DISABLE
5	0x0	VDE4:See VDE0 description
4	0x0	VDE3:See VDE0 description
3	0x0	VDE2:See VDE0 description
2	0x0	VDE1:See VDE0 description
1	0x0	VDE0: This bit is set to '1' only by a write of '1'. All other writes are ignored. Once it is set to '1', it remains '1' and only resets by system reset (same resets which resets main PMC).This bit is send to VDE via a sideband signal.

Bit	Reset	Description
0	0x0	HDA_LPBK_DIS: Sticky one bit to disable the loopback in HDA codec. This bit is set to '1' only by a write of '1'. All other writes are ignored. Once it is set to '1', it remains '1' and only resets by system reset (the same reset which resets the main PMC).

10.9.177 APBDEV_PMC_SEC_DISABLE2_0

Offset: 0x2c4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	READ23: disable read from secure register 23 0 = OFF 1 = ON
30	0x0	WRITE23: disable write to secure register 23 0 = OFF 1 = ON
29	0x0	READ22: disable read from secure register 22 0 = OFF 1 = ON
28	0x0	WRITE22: disable write to secure register 22 0 = OFF 1 = ON
27	0x0	READ21: disable read from secure register 21 0 = OFF 1 = ON
26	0x0	WRITE21: disable write to secure register 21 0 = OFF 1 = ON
25	0x0	READ20: disable read from secure register 20 0 = OFF 1 = ON
24	0x0	WRITE20: disable write to secure register 20 0 = OFF 1 = ON
23	0x0	READ19: disable read from secure register 19 0 = OFF 1 = ON
22	0x0	WRITE19: disable write to secure register 19 0 = OFF 1 = ON
21	0x0	READ18: disable read from secure register 18 0 = OFF 1 = ON
20	0x0	WRITE18: disable write to secure register 18 0 = OFF 1 = ON
19	0x0	READ17: disable read from secure register 17 0 = OFF 1 = ON
18	0x0	WRITE17: disable write to secure register 17 0 = OFF 1 = ON
17	0x0	READ16: disable read from secure register 16 0 = OFF 1 = ON

Bit	Reset	Description
16	0x0	WRITE16: disable write to secure register 16 0 = OFF 1 = ON
15	0x0	READ15: disable read from secure register 15 0 = OFF 1 = ON
14	0x0	WRITE15: disable write to secure register 15 0 = OFF 1 = ON
13	0x0	READ14: disable read from secure register 14 0 = OFF 1 = ON
12	0x0	WRITE14: disable write to secure register 14 0 = OFF 1 = ON
11	0x0	READ13: disable read from secure register 13 0 = OFF 1 = ON
10	0x0	WRITE13: disable write to secure register 13 0 = OFF 1 = ON
9	0x0	READ12: disable read from secure register 12 0 = OFF 1 = ON
8	0x0	WRITE12: disable write to secure register 12 0 = OFF 1 = ON
7	0x0	READ11: disable read from secure register 11 0 = OFF 1 = ON
6	0x0	WRITE11: disable write to secure register 11 0 = OFF 1 = ON
5	0x0	READ10: disable read from secure register 10 0 = OFF 1 = ON
4	0x0	WRITE10: disable write to secure register 10 0 = OFF 1 = ON
3	0x0	READ9: disable read from secure register 9 0 = OFF 1 = ON
2	0x0	WRITE9: disable write to secure register 9 0 = OFF 1 = ON
1	0x0	READ8: disable read from secure register 8 0 = OFF, 1 = ON 0 = OFF 1 = ON
0	0x0	WRITE8: disable write to secure register 8 0 = OFF, 1 = ON 0 = OFF 1 = ON

10.9.178 APBDEV_PMC_WEAK_BIAS_0

Offset: 0x2c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:8	0x0	EMMC
7:6	0x0	XM1_EXCLK
5:4	0x0	XM1_CLK
3:2	0x0	XM0_EXCLK
1:0	0x0	XM0_CLK: weak_bias

10.9.179 APBDEV_PMC_REG_SHORT_0

Offset: 0x2cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	DPD_VAL_XM1_EXCLK: reg_short value during DPD operation for VTTGEN cells of xm1_exclk group
6	0x0	DPD_VAL_XM1_CLK: reg_short value during DPD operation for VTTGEN cells of xm1_clk group
5	0x0	DPD_VAL_XM0_EXCLK: reg_short value during DPD operation for VTTGEN cells of xm0_exclk group
4	0x0	DPD_VAL_XM0_CLK: reg_short value during DPD operation for VTTGEN cells of xm0_clk group
3	0x0	VAL_XM1_EXCLK: reg_short value during normal operation for VTTGEN cells of xm1_exclk group
2	0x0	VAL_XM1_CLK: reg_short value during normal operation for VTTGEN cells of xm1_clk group
1	0x0	VAL_XM0_EXCLK: reg_short value during normal operation for VTTGEN cells of xm0_exclk group
0	0x0	VAL_XM0_CLK: reg_short value during normal operation for vttgen cells of xm0_clk group

10.9.180 APBDEV_PMC_PG_MASK_ANDOR_0

Power Gate Mask AND (for force power-gating) or OR (for force power-ungating) the mask value comes from the corresponding PG_MASK* register

Offset: 0x2d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000x000x0000x00x0x000)

Bit	Reset	Description
22	0x0	XUSBC: Used as AND or OR override for XUSBC. 0 = AND 1 = OR
21	0x0	XUSBB: Used as AND or OR override for XUSBB. 0 = AND 1 = OR
20	0x0	XUSBA: Used as AND or OR override for XUSBA. 0 = AND 1 = OR
19	0x0	DISB: Used as AND or OR override for DISB partition. 0 = AND 1 = OR

Bit	Reset	Description
18	0x0	DIS: Used as AND or OR override for DIS partition. 0 = AND 1 = OR
16	0x0	C1NC: Used as AND or OR override for Cluster 1 Non-CPU partition. 0 = AND 1 = OR
15	0x0	C0NC: Used as AND or OR override for Cluster 0 Non-CPU partition. 0 = AND 1 = OR
14	0x0	CE0: Used as AND or OR override for CPU0 partition. 0 = AND 1 = OR
12	0x0	CELP: Used as AND or OR override for CELP partition. 0 = AND 1 = OR
11	0x0	CE3: Used as AND or OR override for CE3 partition. 0 = AND 1 = OR
10	0x0	CE2: Used as AND or OR override for CE2 partition. 0 = AND 1 = OR
9	0x0	CE1: Used as AND or OR override for CE1 partition. 0 = AND 1 = OR
7	0x0	HEG: Used as AND or OR override for HEG partition. 0 = AND 1 = OR
6	0x0	MPE: Used as AND or OR override for MPE partition. 0 = AND 1 = OR
4	0x0	VDE: Used as AND or OR override for VDE0 partition. 0 = AND 1 = OR
2	0x0	VE: Used as AND or OR override for VE partition. 0 = AND 1 = OR
1	0x0	TD: Used as AND or OR override for TD. 0 = AND 1 = OR Note: The PMC generates sleep enables that are active High (that is, 1 = power on).
0	0x0	RESERVED: 0 = AND 1 = OR

10.9.181 APBDEV_PMC_SECURE_SCRATCH8_0

Secure Scratch Register

Offset: 0x300 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH8

10.9.182 APBDEV_PMC_SECURE_SCRATCH9_0

Secure Scratch Register

Offset: 0x304 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH9

10.9.183 APBDEV_PMC_SECURE_SCRATCH10_0

Secure Scratch Register

Offset: 0x308 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH10

10.9.184 APBDEV_PMC_SECURE_SCRATCH11_0

Secure Scratch Register

Offset: 0x30c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH11

10.9.185 APBDEV_PMC_SECURE_SCRATCH12_0

Secure Scratch Register

Offset: 0x310 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH12

10.9.186 APBDEV_PMC_SECURE_SCRATCH13_0

Secure Scratch Register

Offset: 0x314 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH13

10.9.187 APBDEV_PMC_SECURE_SCRATCH14_0

Secure Scratch Register

Offset: 0x318 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH14

10.9.188 APBDEV_PMC_SECURE_SCRATCH15_0

Secure Scratch Register

Offset: 0x31c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH15

10.9.189 APBDEV_PMC_SECURE_SCRATCH16_0

Secure Scratch Register

Offset: 0x320 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH16

10.9.190 APBDEV_PMC_SECURE_SCRATCH17_0

Secure Scratch Register

Offset: 0x324 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH17

10.9.191 APBDEV_PMC_SECURE_SCRATCH18_0

Secure Scratch Register

Offset: 0x328 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH18

10.9.192 APBDEV_PMC_SECURE_SCRATCH19_0

Secure Scratch Register

Offset: 0x32c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH19

10.9.193 APBDEV_PMC_SECURE_SCRATCH20_0

Secure Scratch Register

Offset: 0x330 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH20

10.9.194 APBDEV_PMC_SECURE_SCRATCH21_0

Secure Scratch Register

Offset: 0x334 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH21

10.9.195 APBDEV_PMC_SECURE_SCRATCH22_0

Secure Scratch Register

Offset: 0x338 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH22

10.9.196 APBDEV_PMC_SECURE_SCRATCH23_0

Secure Scratch Register

Offset: 0x33c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SECURE_SCRATCH23

10.9.197 APBDEV_PMC_SCRATCH56_0

Scratch Register

Offset: 0x340 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH56: General-purpose register storage

10.9.198 APBDEV_PMC_SCRATCH57_0

Scratch Register

Offset: 0x344 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH67: General-purpose register storage

10.9.199 APBDEV_PMC_SCRATCH58_0

Scratch Register

Offset: 0x348 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH58: General-purpose register storage

10.9.200 APBDEV_PMC_SCRATCH59_0

Scratch Register

Offset: 0x34c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH59: General-purpose register storage

10.9.201 APBDEV_PMC_SCRATCH60_0

Scratch Register

Offset: 0x350 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH60: General-purpose register storage

10.9.202 APBDEV_PMC_SCRATCH61_0

Scratch Register

Offset: 0x354 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH61: General-purpose register storage

10.9.203 APBDEV_PMC_SCRATCH62_0

Scratch Register

Offset: 0x358 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH62: General-purpose register storage

10.9.204 APBDEV_PMC_SCRATCH63_0

Scratch Register

Offset: 0x35c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH63: General-purpose register storage

10.9.205 APBDEV_PMC_SCRATCH64_0

Scratch Register

Offset: 0x360 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH64: General-purpose register storage

10.9.206 APBDEV_PMC_SCRATCH65_0

Scratch Register

Offset: 0x364 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH65: General-purpose register storage

10.9.207 APBDEV_PMC_SCRATCH66_0

Scratch Register

Offset: 0x368 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH66: General- purpose register storage

10.9.208 APBDEV_PMC_SCRATCH67_0

Scratch Register

Offset: 0x36c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH67: General-purpose register storage

10.9.209 APBDEV_PMC_SCRATCH68_0

Scratch Register

Offset: 0x370 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH68: General-purpose register storage

10.9.210 APBDEV_PMC_SCRATCH69_0

Scratch Register

Offset: 0x374 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH69: General-purpose register storage

10.9.211 APBDEV_PMC_SCRATCH70_0

Scratch Register

Offset: 0x378 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH70: General-purpose register storage

10.9.212 APBDEV_PMC_SCRATCH71_0

Scratch Register

Offset: 0x37c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH71: General-purpose register storage

10.9.213 APBDEV_PMC_SCRATCH72_0

Scratch Register

Offset: 0x380 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH72: General-purpose register storage

10.9.214 APBDEV_PMC_SCRATCH73_0

Scratch Register

Offset: 0x384 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH73: General-purpose register storage

10.9.215 APBDEV_PMC_SCRATCH74_0

Scratch Register

Offset: 0x388 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH74: General-purpose register storage

10.9.216 APBDEV_PMC_SCRATCH75_0

Scratch Register

Offset: 0x38c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH75: General-purpose register storage

10.9.217 APBDEV_PMC_SCRATCH76_0

Scratch Register

Offset: 0x390 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH76: General-purpose register storage

10.9.218 APBDEV_PMC_SCRATCH77_0

Scratch Register

Offset: 0x394 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH77: General-purpose register storage

10.9.219 APBDEV_PMC_SCRATCH78_0

Scratch Register

Offset: 0x398 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH78: General-purpose register storage

10.9.220 APBDEV_PMC_SCRATCH79_0

Scratch Register

Offset: 0x39c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH79: General-purpose register storage

10.9.221 APBDEV_PMC_SCRATCH80_0

Scratch Register

Offset: 0x3a0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH80: General-purpose register storage

10.9.222 APBDEV_PMC_SCRATCH81_0

Scratch Register

Offset: 0x3a4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH81: General-purpose register storage

10.9.223 APBDEV_PMC_SCRATCH82_0

Scratch Register

Offset: 0x3a8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH82: General-purpose register storage

10.9.224 APBDEV_PMC_SCRATCH83_0

Scratch Register

Offset: 0x3ac | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH83: General-purpose register storage

10.9.225 APBDEV_PMC_SCRATCH84_0

Scratch Register

Offset: 0x3b0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH84: General-purpose register storage

10.9.226 APBDEV_PMC_SCRATCH85_0

Scratch Register

Offset: 0x3b4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH85: General-purpose register storage

10.9.227 APBDEV_PMC_SCRATCH86_0

Scratch Register

Offset: 0x3b8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH86: General-purpose register storage

10.9.228 APBDEV_PMC_SCRATCH87_0

Scratch Register

Offset: 0x3bc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH87: General-purpose register storage

10.9.229 APBDEV_PMC_SCRATCH88_0

Scratch Register

Offset: 0x3c0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH88: General-purpose register storage

10.9.230 APBDEV_PMC_SCRATCH89_0

Scratch Register

Offset: 0x3c4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH89: General-purpose register storage

10.9.231 APBDEV_PMC_SCRATCH90_0

Scratch Register

Offset: 0x3c8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH90: General-purpose register storage

10.9.232 APBDEV_PMC_SCRATCH91_0

Scratch Register

Offset: 0x3cc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH91: General-purpose register storage

10.9.233 APBDEV_PMC_SCRATCH92_0

Scratch Register

Offset: 0x3d0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH92: General-purpose register storage

10.9.234 APBDEV_PMC_SCRATCH93_0

Scratch Register

Offset: 0x3d4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH93: General-purpose register storage

10.9.235 APBDEV_PMC_SCRATCH94_0

Scratch Register

Offset: 0x3d8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH94: General-purpose register storage

10.9.236 APBDEV_PMC_SCRATCH95_0

Scratch Register

Offset: 0x3dc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH95: General-purpose register storage

10.9.237 APBDEV_PMC_SCRATCH96_0

Scratch Register

Offset: 0x3e0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH96: General-purpose register storage

10.9.238 APBDEV_PMC_SCRATCH97_0

Scratch Register

Offset: 0x3e4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH97: General-purpose register storage

10.9.239 APBDEV_PMC_SCRATCH98_0

Scratch Register

Offset: 0x3e8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH98: General-purpose register storage

10.9.240 APBDEV_PMC_SCRATCH99_0

Scratch Register

Offset: 0x3ec | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH99: General-purpose register storage

10.9.241 APBDEV_PMC_SCRATCH100_0

Scratch Register

Offset: 0x3f0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH100: General-purpose register storage

10.9.242 APBDEV_PMC_SCRATCH101_0

Scratch Register

Offset: 0x3f4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH101: General-purpose register storage

10.9.243 APBDEV_PMC_SCRATCH102_0

Scratch Register

Offset: 0x3f8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH102: General-purpose register storage

10.9.244 APBDEV_PMC_SCRATCH103_0

Scratch Register

Offset: 0x3fc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH103: General-purpose register storage

10.9.245 APBDEV_PMC_SCRATCH104_0

Scratch Register

Offset: 0x400 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH104: General-purpose register storage

10.9.246 APBDEV_PMC_SCRATCH105_0

Scratch Register

Offset: 0x404 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH105: General-purpose register storage

10.9.247 APBDEV_PMC_SCRATCH106_0

Scratch Register

Offset: 0x408 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH106: General-purpose register storage

10.9.248 APBDEV_PMC_SCRATCH107_0

Scratch Register

Offset: 0x40c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH107: General-purpose register storage

10.9.249 APBDEV_PMC_SCRATCH108_0

Scratch Register

Offset: 0x410 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH108: General-purpose register storage

10.9.250 APBDEV_PMC_SCRATCH109_0

Scratch Register

Offset: 0x414 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH109: General-purpose register storage

10.9.251 APBDEV_PMC_SCRATCH110_0

Scratch Register

Offset: 0x418 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH110: General-purpose register storage

10.9.252 APBDEV_PMC_SCRATCH111_0

Scratch Register

Offset: 0x41c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH111: General-purpose register storage

10.9.253 APBDEV_PMC_SCRATCH112_0

Scratch Register

Offset: 0x420 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH112: General-purpose register storage

10.9.254 APBDEV_PMC_SCRATCH113_0

Scratch Register

Offset: 0x424 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH113: General-purpose register storage

10.9.255 APBDEV_PMC_SCRATCH114_0

Scratch Register

Offset: 0x428 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH114: General-purpose register storage

10.9.256 APBDEV_PMC_SCRATCH115_0

Scratch Register

Offset: 0x42c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH115: General-purpose register storage

10.9.257 APBDEV_PMC_SCRATCH116_0

Scratch Register

Offset: 0x430 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH116: General-purpose register storage

10.9.258 APBDEV_PMC_SCRATCH117_0

Scratch Register

Offset: 0x434 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH117: General-purpose register storage

10.9.259 APBDEV_PMC_SCRATCH118_0

Scratch Register

Offset: 0x438 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH118: General-purpose register storage

10.9.260 APBDEV_PMC_SCRATCH119_0

Scratch Register

Offset: 0x43c │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SCRATCH119: General-purpose register storage

10.10 PMC Counter 0 Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

10.10.1 SYSCTR0_CNTCR_0

The control registers are read/written by secure accesses only except the CNTFID0 and COUNTERID11-0 registers, which can be written only once by secure or non-secure access.

The CNTFID0 and COUNTERID11-0 registers are read-only; however, to initialize them at boot (by ARM7 or main CPU), they are defined as write once.

A non-secure write to the control registers is ignored. A non-secure read to the control registers returns all 1s.

Counter Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx00)

Bit	Reset	Description
8	0x0	FCREQ: Requested frequency modes table entry. This bit is not used.
1	0x0	HDBG: Halt-on-debug. Controls whether a Halt-on-debug signal halts the system counter or not. 0: System counter ignores Halt-on-debug.1: Asserted Halt-on-debug signal halts the system counter update. 0 = DISABLE 1 = ENABLE
0	0x0	EN: Enables the counter. 0: System counter disabled. 1: System counter enabled. When this bit is written to '1', then the TSC is loaded with the CNTCVx register’s value, and starts incrementing from that value. When this bit is written to '0', the TSC halts counting and stays at that value. 0 = DISABLE 1 = ENABLE

10.10.2 SYSCTR0_CNTSR_0

Counter Status Register

Offset: 0x4 | Read/Write: RO | Reset: 0x00000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	FCREQ: Frequency change acknowledge. This bit is a copy of the value of the CNTCR[FCREQ] bit.
1	X	HDBG: Indicates whether or not the counter is halted because the Halt-on-Debug signal is asserted: 0: Counter is not halted.1: Counter is halted. 0 = DISABLE 1 = ENABLE
0	X	RESERVED: Reserved, no impact.

10.10.3 SYSCTR0_CNTCV0_0

Counter Count Value [31:0] Register

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

Bit	Reset	Description
31:0	0x0	CV: Counter value [31:0]. A read of this register provides the TSC[31:0] value at the time of the read. When CNTCR[EN]=0, a write to this register is used to initialize the TSC[31:0] value. When CNTCR[EN]=1, a write to this register has an unpredictable behavior.

10.10.4 SYSCTR0_CNTCV1_0

Counter Count Value[63:32] Register

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

Bit	Reset	Description
31:0	0x0	CV: Counter value [63:32]. A read of this register provides the TSC[63:32] value at the time of the read. When CNTCR[EN]=0, a write to this register is used to initialize the TSC[63:32] value. When CNTCR[EN]=1, a write to this register has an unpredictable behavior.

10.10.5 SYSCTR0_CNTFID0_0

Frequency Table Entry Register

Offset: 0x20 | Read/Write: R/W | Reset: 0x00b71b00 (0b00000000101101110001101100000000)

Bit	Reset	Description
31:0	0xb71b00	FV: Counter frequency value in Hz. The default is set to 12 MHz. This register can be written only once.

10.10.6 SYSCTR0_CNTFID1_0

Frequency Table End Marker

Offset: 0x24 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	FV: Counter frequency value end marker (all-0s, read-only).

10.10.7 SYSCTR0_COUNTERID4_0

The COUNTERID11-0 registers are read-only; however, to initialize them at boot (by ARM7 or main CPU), they are defined as write once.

Offset: 0xfd0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	VALUE: Peripheral ID value

10.10.8 SYSCTR0_COUNTERID5_0

Offset: 0xfd4 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	RESERVED: Reserved. A read will return all 0s.

10.10.9 SYSCTR0_COUNTERID6_0

Offset: 0xfd8 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	RESERVED: Reserved. A read will return all 0s.

10.10.10 SYSCTR0_COUNTERID7_0

Offset: 0xfdc | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	RESERVED: Reserved. A read will return all 0s.

10.10.11 SYSCTR0_COUNTERID0_0

Offset: 0xfe0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	VALUE: Peripheral ID value

10.10.12 SYSCTR0_COUNTERID1_0

Offset: 0xfe4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	VALUE: Peripheral ID value

10.10.13 SYSCTR0_COUNTERID2_0

Offset: 0xfe8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	VALUE: Peripheral ID value

10.10.14 SYSCTR0_COUNTERID3_0

Offset: 0xfec | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	VALUE: Peripheral ID value

10.10.15 SYSCTR0_COUNTERID8_0

Offset: 0xff0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	VALUE: Component ID value

10.10.16 SYSCTR0_COUNTERID9_0

Offset: 0xff4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	VALUE: Component ID value

10.10.17 SYSCTR0_COUNTERID10_0

Offset: 0xff8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	VALUE: Component ID value

10.10.18 SYSCTR0_COUNTERID11_0

Offset: 0xffc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	VALUE: Component ID value

10.11 PMC Counter 1 Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

The status registers are some control registers readable via the CNTReadBase base address by secure or non-secure accesses.

These are the same physical registers that are described in the Control Register section.

10.11.1 SYSCTR1_CNTCV0_0

Counter Count Value [31:0] Register

Offset: 0x8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CV: Counter value [31:0]

10.11.2 SYSCTR1_CNTCV1_0

Counter Count Value [63:32] Register

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

Bit	Reset	Description
31:0	X	CV: Counter value [63:32]

10.11.3 SYSCTR1_COUNTERID4_0

Offset: 0xfd0 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	VALUE: Peripheral ID value.

10.11.4 SYSCTR1_COUNTERID5_0

Offset: 0xfd4 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	RESERVED: Reserved. A read will return all 0s.

10.11.5 SYSCTR1_COUNTERID6_0

Offset: 0xfd8 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	RESERVED: Reserved. A read will return all 0s.

10.11.6 SYSCTR1_COUNTERID7_0

Offset: 0xfdc | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	RESERVED: Reserved. A read will return all 0s.

10.11.7 SYSCTR1_COUNTERID0_0

Offset: 0xfe0 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	VALUE: Peripheral ID value

10.11.8 SYSCTR1_COUNTERID1_0

Offset: 0xfe4 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	VALUE: Peripheral ID value

10.11.9 SYSCTR1_COUNTERID2_0

Offset: 0xfe8 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	VALUE: Peripheral ID value

10.11.10 SYSCTR1_COUNTERID3_0

Offset: 0xfec | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	VALUE: Peripheral ID value

10.11.11 SYSCTR1_COUNTERID8_0

Offset: 0xff0 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	VALUE: Component ID value

10.11.12 SYSCTR1_COUNTERID9_0

Offset: 0xff4 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	VALUE: Component ID value

10.11.13 SYSCTR1_COUNTERID10_0

Offset: 0xff8 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:0	X	VALUE: Component ID value

10.11.14 SYSCTR1_COUNTERID11_0

Offset: 0xffc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	VALUE: Component ID value

[THIS PAGE INTENTIONALLY LEFT BLANK]

11.0 REAL-TIME CLOCK

The Real-Time Clock (RTC) module maintains seconds and milliseconds counters, and five alarm registers. The RTC is in the ‘always-on’ power domain, allowing for the counters to run and alarms to trigger when the system is in low-power state. If configured, interrupts triggered by the RTC can cause the system to wake up from a low-power state.

Features

- 10-bit milliseconds counter that runs off of a 32.768 KHz clock source.
- 32-bit seconds counter that increments for every 1000 milliseconds.
- Alarm feature that triggers an interrupt when the specified value matches the milliseconds counter.
- Alarm feature that triggers an interrupt when the specified value matches the seconds counter.
- Count-down alarm feature that triggers an alarm after counting down the specified number of seconds.
- Count-down alarm feature that triggers an alarm after counting down the specified number of milliseconds.
- Security bit that disables further processor writes to the seconds counter and ensures that the RTC clock keeps running.
- Hardware adjusts drift in clock which can occur due to ppm variations in oscillator output.

11.1 Functional Description

The RTC operates in two clock domains: the APB clock domain and the 32 KHz clock domain. The RTC continues updating the milliseconds and seconds counters and continues triggering interrupts even when the MAIN partition is powered down. Since the APB clock is disabled when MAIN is powered down, registers are implemented in the 32 KHz clock domain.

All the registers except the BUSY register are implemented in the 32 KHz clock domain. Writes are transferred to the 32 KHz domain with BUSY.STATUS set to BUSY until the transfer is completed. Reads are shadowed in the APB clock domain and return immediately.

The RTC implements a Seconds Counter register, three Alarm registers, two countdown alarms, and various interrupt related registers.

Writes to the seconds counter can be disabled by writing to the CONTROL.DIS_WR_SEC_CNT bit. Alarm registers and countdown alarm registers set interrupt bits when corresponding events occur. Interrupt status, mask, set and source registers reflect the status and also allow setting and clearing of various bits. All registers (except the BUSY register) are implemented in the 32K clock domain.

Resets

The RTC receives an asynchronous reset which is synchronized to the APB clock and RTC clock domains.

11.2 RTC Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

11.2.1 APBDEV_RTC_CONTROL_0

The RTC implements a Seconds Counter register, three Alarm registers, two countdown alarms, and various interrupt related registers. Writes to the seconds counter can be disabled by writing to the CONTROL.DIS_WR_SEC_CNT bit.

Alarm registers and countdown alarm registers set interrupt bits when corresponding events occur. Interrupt status, mask, set and source registers reflect the status as well as allow setting and clearing of various bits.

All the registers except the BUSY registers are implemented in the 32K clock domain. Writes are transferred to the 32K domain with BUSY STATUS set to BUSY until the transfer completes. Reads are shadowed in the APB domain and will return immediately.

Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	WR_SEC_CNT: When set, writes to the SECONDS counter are disabled. Can only be cleared by resetting the RTC module 0 = DISABLE 1 = ENABLE

11.2.2 APBDEV_RTC_BUSY_0

Busy Register

Offset: 0x4 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	STATUS: This bit is set when a write is initiated on the APB side. It is cleared once the write completes in RTC 32KHz clock domain, which could be several thousands of APB clocks. This must be IDLE before a write is initiated. Note that this bit is only for writes. 0 = IDLE 1 = BUSY

11.2.3 APBDEV_RTC_SECONDS_0

Seconds Counter Register

SECONDS register is copied over to APB side every eight 32 KHz clocks (~250 µs). Because of this, performing a read immediately after a write might return old value. This covers 49710.26 Days (or) 136.192 Years of 365 days each.

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

Bit	Reset	Description
31:0	0x0	SECONDS: The seconds counter is incremented for every 1000 milliseconds

11.2.4 APBDEV_RTC_SHADOW_SECONDS_0

Shadowed Seconds Counter Register

Shadow SECONDS register is updated over to the APB side whenever there is a read to milliseconds counter.

Since the software cannot read both registers at any given point of time, the Seconds register content is captured in this register. If software needs to read both seconds and milliseconds, read the MILLI_SECONDS register followed by a read of this register.

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

Bit	Reset	Description
31:0	X	SHADOW_SECONDS: A snapshot of the SECONDS counter is taken, whenever there is a read to MILLI_SECONDS Register.

11.2.5 APBDEV_RTC_MILLI_SECONDS_0

Milliseconds Counter Register

The Milliseconds register is copied over to APB side every eight 32 KHz clocks (~250 μs). Because of this, performing a read immediately after a write might return an old value.

Offset: 0x10 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	MILLI_SECONDS: Milliseconds counter is incremented using the Bresenham algorithm

11.2.6 APBDEV_RTC_SECONDS_ALARM0_0

Seconds Alarm0 Registers

When the value in this register matches the seconds counter, corresponding interrupt status bit is set. If enabled, the interrupt line is asserted. A value of zero disables the alarm.

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECS_MATCH_VALUE: Match value to trigger the alarm

11.2.7 APBDEV_RTC_SECONDS_ALARM1_0

Seconds Alarm1 Registers

When the value in this register matches the seconds counter, corresponding interrupt status bit is set. If enabled, the interrupt line is asserted. A value of zero disables the alarm.

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SECS_MATCH_VALUE: Match value to trigger the alarm

11.2.8 APBDEV_RTC_MILLI_SECONDS_ALARM_0

Milliseconds Alarm Register

When the value in this register matches the milliseconds counter, corresponding interrupt status bit is set. If enabled, the interrupt line is asserted. A value of zero disables the alarm.

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	MSEC_MATCH_VALUE: milliseconds match value.

11.2.9 APBDEV_RTC_SECONDS_COUNTDOWN_ALARM_0

Countdown Alarm Registers

If ENABLE is set to ENABLED, an internal counter is loaded with VALUE and counted down. The interrupt bit corresponding to the countdown_alarm_0 is set after the specified number of seconds have elapsed.

The interrupt bit corresponding to the countdown alarm (MSC_CDN_ALARM, SEC_CDN_ALARM) is set after the specified interval has expired. If REPEAT is ENABLED, the countdown operation is performed repeatedly.

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	ENABLE: enable bit for the countdown operation. If repeat is not set, this bit is cleared once the internal counters counts down to specified value. 0 = DISABLED 1 = ENABLED
30	0x0	REPEAT: repeat bit for the countdown operation 0 = DISABLED 1 = ENABLED
29:0	0x0	VALUE: number of seconds to countdown

11.2.10 APBDEV_RTC_MILLI_SECONDS_COUNTDOWN_ALARM_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	ENABLE: enable bit for the countdown operation. If repeat is not set, this bit is cleared once the internal counters counts down to specified value. 0 = DISABLED 1 = ENABLED
30	0x0	REPEAT: repeat bit for the countdown operation 0 = DISABLED 1 = ENABLED
29:0	0x0	VALUE: number of milliseconds to countdown

11.2.11 APBDEV_RTC_INTR_MASK_0

Interrupt Mask Register

This register stores the masks for the interrupts. If a bit is set 1 and the corresponding interrupt condition is satisfied, interrupt line to system interrupt controller is asserted.

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

11.2.12 APBDEV_RTC_INTR_STATUS_0

Interrupt Status Register

Bits in this register are high after the interrupt condition is satisfied. A write to this register clears the bits corresponding to the data bits that are high in the write data.

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

11.2.13 APBDEV_RTC_INTR_SOURCE_0

Interrupt Source Register

This read-only register returns the AND of the Interrupt Source and Interrupt Mask registers.

Offset: 0x30 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
4	X	MSEC_CDN_ALARM
3	X	SEC_CDN_ALARM
2	X	MSEC_ALARM
1	X	SEC_ALARM1
0	X	SEC_ALARM0

11.2.14 APBDEV_RTC_INTR_SET_0

Interrupt Set Register

This write-only register can be used to set the interrupt status bit. A write to this register causes the bits in status register to be set if the corresponding bit in write data is 1'b1. Reads always return 'h0.

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	MSEC_CDN_ALARM
3	0x0	SEC_CDN_ALARM
2	0x0	MSEC_ALARM
1	0x0	SEC_ALARM1
0	0x0	SEC_ALARM0

11.2.15 APBDEV_RTC_CORRECTION_FACTOR_0

Correction Factor (Digital Trimming) Register

Support is for +/- 500ppm, but +/- 50 ppm is maximum

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	DIRECTION: 0 = DECREMENT 1 = INCREMENT
8:0	0x0	PPM

12.0 BOOT PROCESS

This section provides a functional description of the boot process.

Glossary and Acronyms

Table 31: Boot ROM Glossary

Acronym	Description
AES	Advanced Encryption Standard
BCT	Boot Configuration Table
BKV	Best Known Value. Configuration determined by system characterization.
BL	Boot Loader, the software that initiates loading the OS
BR	Boot ROM, the on-chip software that is executed first after reset.
COP	COProcessor (also known as AVP or ARM7)
FA	Failure Analysis
IRAM	Internal RAM used by the boot process until DRAM is configured.
IROM	Internal ROM which contains the Boot ROM code and data.
LP0	Low-Power 0 sleep state.
LX	LP0 Exit code
MB	Micro Boot, an optional second stage of the boot process
PKC	Public-Key Cryptography
RCM	Recovery Mode, used when the system is unable to boot normally
SE	Security Engine
WoA	Windows-on-ARM.

12.1 Boot Components

Table 32: Boot Components

Component	Execution	Stored Location	Run Location	Target	Ownership
Boot ROM	AVP	IROM	IROM	64KB	NVIDIA
BCT	n/a	Media	IRAM	4KB	OEM
MicroBoot	AVP	Media	IRAM	~175KB	NVIDIA + ODM
Boot Loader	AVP + CPU	Media	DRAM	Same	ODM
LP0 Exit Code	AVP	DRAM	DRAM	Same	NVIDIA
OS	CPU	Media	DRAM	Same	ODM
TSEC RNG	Falcon (µC)	Media	µC RAM	~4KB	NVIDIA
TSEC FW*	Falcon (µC)	Media	µC RAM		NVIDIA
MSENC FW*	Falcon (µC)	Media	µC RAM		NVIDIA

* Not required for boot

12.1.1 Boot ROM

The very first code to execute upon reset is the Boot ROM. It is contained in an on-chip 64KB IROM, executes completely on the AVP, and is the very first firmware to execute upon chip reset. It is hardcoded into the chip and is not reprogrammable. The Boot ROM represents the root-of-trust for boot. Its goal is to be as simple and robust as possible. The Boot ROM must be able to locate, configure, and read from the boot media based on fuses and hardware straps.

Before handoff to the MicroBoot or Boot Loader, the Boot ROM must restrict access to any sensitive information including the AES keys. Note that after lock down the keys are unreadable by software but remains present within key slots of the SE, and can be used for authentication and decryption by subsequent stages of boot.

12.1.2 Boot Configuration Table (BCT)

The Boot Configuration Table is read initially by the Boot ROM from the boot media, is approximately 4KB in size, and contains platform-specific configuration information to be used by the boot process. One important piece of information in the BCT is the DRAM geometry and timing information.

The BCT stores information needed to locate and verify a boot loader.

NvBootLoaderInfo Structure

There is one NvBootLoaderInfo structure for each copy of a BL stored on the device.

Table 33: NvBootLoaderInfoRec Structure in the BCT

Parameter Name	Type	Description
Attribute	NvU32	Specifies an attribute available for use by other code. Not interpreted by the Boot ROM.
EntryPoint	NvU32	Specifies the entry point address in the loaded BL image.
Length	NvU32	Specifies the length of the BL in bytes. BLs must be padded to an integral number of 16 bytes with the padding pattern. Note The end of the BL cannot fall within the last 16 bytes of a page. Add another 16 bytes to work around this restriction if needed.
LoadAddress	NvU32	Specifies the starting address of the memory region into which the BL will be loaded.
Signature	NvBootObjectSignature	Specifies the AES-CMAC MAC or RSASSA-PSS signature of the BL.
StartBlock	NvU32	Specifies the first physical block on the secondary boot device that contains the start of the BL. The first block can never be a known bad block.
StartPage	NvU32	Specifies the page within the first block that contains the start of the BL.
Version	NvU32	Specifies a version number for the BL. The assignment of numbers is arbitrary; the numbers are only used to identify redundant copies (which have the same version number) and to distinguish between different versions of the BL (which have different numbers).

The following table lists the enumerated types in the BCT that identify the devices from which the system booted.

Note that these no longer match the fuse API device values (for backward compatibility with legacy devices).

Table 34: Enumerated Types in the BCT

Type	Description
NvBootDevType_None	Specifies a default (unset) value. Set to 0.
NvBootDevType_Reserved	Reserved
NvBootDevType_Spi	Specifies the SPI NOR
NvBootDevType_Sdmmc	Specifies the SDMMC
NvBootDevType_Reserved	Reserved
NvBootDevType_Reserved	Reserved

Type	Description
NvBootDevType_Max	
NvBootDevType_Force32	Set to 0x7FFFFFFF

The NvBootConfigTableRec structure contains the information needed to load BLs from the secondary boot device.

- Supplying NumParamSets = 0 indicates not to load any of them.
- Supplying NumDramSets = 0 indicates not to load any of them.
- The RandomAesBlock member exists to increase the difficulty of key attacks based on knowledge of this structure.

Table 35: NvBootConfigTableRec Structure

Parameter Name	Type	Description
BadBlockTable	NvBootBadBlockTable	Specifies the bad block table, which is defined in nvboot_badblocks.h
Key	NvBootRsaKeyModulus	Specifies the RSA public key's modulus
Signature	NvBootObjectSignature	Specifies the AES-CMAC MAC or RSASSA-PSS signature for the rest of the BCT structure
CustomerData[NVBOOT_BCT_CUSTOMER_DATA_SIZE]	NvU8	Specifies a region of data available to customers of the BR. This data region is primarily used by a manufacturing utility or BL to store useful information that needs to be shared among manufacturing utility, BL, and OS image. The BR only provides framework and does not use this data
Signed Section of the BCT		
RandomAesBlock	NvBootHash	Specifies a chunk of random data
UniqueChipId	NvBootECID	Specifies the Unique ID / ECID of the chip that this BCT is specifically generated for. This field is required if SecureJtagControl == NV_TRUE. It is optional otherwise. This is to prevent a signed BCT with SecureJtagControl == NV_TRUE being leaked into the field that would enable JTAG debug for all devices signed with the same private RSA key
BootDataVersion	NvU32	Specifies the version of the BR data structures used to build this BCT. BootDataVersion must match the version number in the BR.
BlockSizeLog2	NvU32	Specifies the size of a physical block on the secondary boot device in log2(bytes)
PageSizeLog2	NvU32	Specifies the size of a page on the secondary boot device in log2(bytes)
PartitionSize	NvU32	Specifies the size of the boot partition in bytes. Used for internal error checking. BLs must fit within this region.

Parameter Name	Type	Description
NumParamSets	NvU32	Specifies the number of valid device parameter sets provided within this BCT. If the device straps are left floating, the same parameters should be replicated to all NVBOOT_BCT_MAX_PARAM_SETS sets.
DevType[NVBOOT_BCT_MAX_PARAM_SETS]	NvBootDevType	Specifies the type of device for parameter set DevParams[i].
DevParams[NVBOOT_BCT_MAX_PARAM_SETS]	NvBootDevParams	Specifies the device parameters with which to reinitialize the secondary boot device controller. The device straps index into this table. The definition of NvBootDevParams is contained within nvboot_devparams.h and the specific device nvboot_*_param.h files.
NumSdramSets	NvU32	Specifies the number of valid SDRAM parameter sets provided within this BCT. If the SDRAM straps are left floating, the same parameters should be replicated to all NVBOOT_BCT_MAX_SDRAM_SETS sets.
SdramParams[NVBOOT_BCT_MAX_SDRAM_SETS]	NvBootSdramParams	Specifies the SDRAM parameters with which to initialize the SDRAM controller. The SDRAM straps index into this table. The definition of NvBootDevParams is contained within nvboot_sdram_param.h.
BootLoadersUsed	NvU32	Specifies the number of BLs described in the BootLoader table
BootLoader[NVBOOT_MAX_BOOTLOADERS]	NvBootLoaderInfo	Specifies the information needed to locate and validate each BL. The BR uses entries 0 through BootLoadersUsed - 1
EnableFailBack	NvBool	Specifies whether FailBack should be used when looking for a good BL.
SecureJtagControl	NvBool	Specify whether or not to enable JTAG access when the JTAG disable fuse has not been burned. SecureJtagControl = NV_FALSE (0) = Disable JTAG access. SecureJtagControl = NV_TRUE (1) = Enable JTAG access.
Reserved[NVBOOT_BCT_RESERVED_SIZE]	NvU8	Specifies a reserved area at the end of the BCT that must be filled with the padding pattern.

12.1.3 Partition Table (PT)

12.1.3.1 Partition Allocation

Generally, a partition is specified in terms of a set of desired logical properties. The underlying physical properties might differ from the logical properties, for example, to hide the presence of bad blocks on the storage media.

The allocation policy governs how the logical partition properties are mapped to the physical partition properties when the partition is first created on the storage device.

The supported allocation properties are:

- Relative--The current partition is placed immediately after the previously defined partition on the storage device. If no previous partition has been defined on the storage device, the partition is placed at physical block zero.
- Absolute--The current partition is placed at the specified physical address on the storage device. The address must be aligned to block (erase) boundaries on the storage device.

For both policies, the client specifies the logical size of the partition in bytes. If the storage device contains any bad blocks when the partition is created, additional blocks will be allocated to the partition such that there will be enough good blocks to store the number of data bytes specified by the client. These good blocks are referred to as logical blocks.

To deal with the possibility that some good blocks might subsequently go bad in the operation, the client can also specify that additional good blocks be allocated to the partition. Depending on the device driver, it might be possible to use these additional good blocks to replace blocks that go bad. Thus, the additional blocks are referred to as *replacement blocks*.

The client specifies the number of replacement blocks as a percentage of the logical blocks. The calculation of replacement blocks is always rounded up to ensure that the client never gets fewer replacement blocks (percentage-wise) than requested.

The calculations for logical blocks and replacement blocks are as follows:

- LB = ceiling(LogicalPartitionSize/BlockSize)
- RB = ceiling(LB * PartitionPercentReserved / 100)

Where RB is the number of reserved blocks, and LB is the number of logical blocks. The block driver for the storage device knows the device's BlockSize attribute and hence can carry out the calculations.

Finally, there is a partition attribute that is passed to the device driver when the partition is created. Interpretation of this attribute is determined by the device driver. This provides a mechanism for associating driver-specific properties with the partition on a per-partition basis.

12.1.3.2 File System Attribute and Strings

There is one file system attribute used by the Partition Manager:

NvPartitionAttr_WriteProtect - 0x1

All strings are in UTF-8 format and have maximum lengths, as defined below.

- If NV_EMBEDDED_BUILD, NVPARTMGR_PARTITION_NAME_LENGTH = 21, otherwise NVPARTMGR_PARTITION_NAME_LENGTH = 4
- NVPARTMGR_MOUNTPATH_NAME_LENGTH = (NVPARTMGR_PARTITION_NAME_LENGTH)

12.1.3.3 Structures

NvFsMountInfoRec Structure

This structure specifies Fstab-like information for a partition. It identifies the location of the partition, the file system type for the partition, and the location (in the file system address space) where the partition is to be mounted.

* The typical process for mounting a partition is as follows:

> **Note:** Information about the size and location of the partition itself must be queried from the partition manager. For an example, see NvPartMgrGetPartInfo().

- Open the device driver for the storage device containing the partition based on the DeviceId, DeviceInstance, and DeviceAttr attributes.
- Open the file system driver for the partition based on the FileSystemType and FileSystemAttr attributes.

- Register the MountPath attribute in the global file system address space such that operations within the MountPath branch of the address space are directed to this instance of the file system driver.

The DeviceAttr and FileSystemAttr attributes are associated with the partition and are used during the mounting process above. They are used to tailor the behavior of the device driver and file system driver on a per-partition basis.

Table 36: NvFsMountInfoRec Structure

Parameter Name	Type	Description
DeviceId	NvU32	Holds the type of device on which partition is located.
DeviceInstance	NvU32	Holds the device instance on which partition is located.
DeviceAttr	NvU32	Not supported.
MountPath[NVPARTMGR_MOUNTPATH_NAME_LENGTH	UTF-8	Holds the path where partition will be mounted in the file system. WARNING: The current implementation requires PartitionName and MountPath to be identical.
FileSystemType	NvU32	Holds the type of file system to mount on this partition.
FileSystemAttr	NvU32	Holds the attribute passed to the file system driver when it is mounted on the partition. The file system driver determines the meaning of this value

NvPartInfoRec Structure

This structure provides MBR-like information for a partition.

Table 37: NvPartInfoRec Structure

Parameter Name	Type	Description
PartitionAttr	NvU32	Not supported.
Padding1	NvU32	Dummy bytes added to avoid implicit padding done on the device side for the PartitionAttr attribute based on the maximum data type of the structure (in this case, NvU64)
StartLogicalSectorAddress	NvU64	Holds the logical start address for the partition, in sectors (of the device where the partition is located).
NumLogicalSectors	NvU64	Holds the logical size of the partition in sectors (of the device where the partition is located).
StartPhysicalSectorAddress	NvU64	Holds the partition start physical sector on the storage media.
EndPhysicalSectorAddress	NvU64	Holds the partition end physical sector on the storage media (inclusive).
PartitionType	NvPartMgrPartitionType	Deprecated.

Parameter Name	Type	Description
Padding2	NvU32	If NV_EMBEDDED_BUILD, Dummy bytes added to avoid implicit padding done on the device side for the PartitionType attribute based on the maximum data type of the structure (in this case, NvU64).
IsWriteProtected	NvU32	If not NV_EMBEDDED_BUILD, Specifies if the partition is write-protected on boot.

NvPartitionStatRec Structure

This structure specifies public partition information.

Table 38: NvPartitionStatRec Structure

Parameter Name	Type	Description
PartitionSize	NvU64	Holds the partition size in bytes.

NvPartAllocInfoRec Structure

This structure defines the allocation policy.

Table 39: NvPartAllocInfoRec Structure

Parameter Name	Type	Description
AllocPolicy	NvPartMgrAllocPolicyType	The allocation policy governing the layout of the partition on the storage device.
StartAddress	NvU64	Physical start address of partition on the storage device, in bytes (must be block aligned). WARNING: This parameter is ignored except when AllocPolicy is NvPartMgrAllocPolicyType_Absolute.
Size	NvU64	The partition is guaranteed to hold the specified amount of data when it is created. The storage capacity of the partition may degrade over time if any of its data blocks fail afterward.
PercentReserved	NvU32	Number of reserved blocks to allocate in the partition, specified as a percentage of the number of logical blocks in the partition. PartitionPercentReserved must be an integer in the range [0, 100]. Calculations are rounded up to guarantee that the actual percentage of reserved blocks allocated meets or exceeds the caller's requirements
AllocAttribute	NvU32	Driver-dependent partition attribute. This value is passed to the relevant device driver when the partition is created. Its meaning is determined by the device driver.
IsWriteProtected	NvU32	If NV_EMBEDDED_BUILD, a flag indicating whether the partition is write-protected. On boot, this attribute is checked, and, if it is TRUE, the primary boot loader write protects the sectors corresponding to this partition.

12.1.3.4 Enumerated Types

The following table listed the enumerated types in the PT.

Table 40: Enumerated Types in the PT

Type	Description
Alignment Policy	
NvPartMgrAllocPolicyType_Absolute	Partition starts at a specified physical address; address must be block-aligned
NvPartMgrAllocPolicyType_Relative	Partition starts at the physical block address immediately following the end of the previous partition
NvPartMgrAllocPolicyType_Force32	Set to 0x7FFFFFFF.
Partition Attributes	
NvPartMgrPartitionAttribute_Preserve	Data should be backed up and then restored.
NvPartMgrPartitionAttribute_Force32	Set to 0x7FFFFFFF.

12.1.4 MicroBoot

MicroBoot may be optionally used by some customers for requirement such as single-boot-image to support multiple DRAM devices, DRAM training, special dead-battery-charging support, improved boot latency, etc. MicroBoot is read and invoked by Boot ROM from the boot media, executes completely on the AVP, and is invoked by the Boot ROM. MicroBoot is responsible for configuring and training DRAM and must itself run from IRAM until then. MicroBoot has combined ownership. It contains NVIDIA-provided boot functionality as well as substantial ODM-specific configuration code.

The Tegra 4 processor Boot adds more functionality and yet targets improved total cold boot latency. MicroBoot can be responsible for improving latency, by moving functionality from the Boot ROM where it was optimized for size and simplicity, to MicroBoot where it can be optimized for performance and parallelism.

12.1.5 Falcon Firmware

The MSENC and TSEC hardware units contain an NVIDIA proprietary Falcon microcontroller, and therefore require their own firmware (FW) to be loaded in order to function. The MSENC and TSEC driver FW, which are not required for boot, are loaded by their OS drivers. However, TSEC RNG FW is required to generate secure key during boot, which is shared with video decode engine. This FW needs to be loaded by secure boot (either by Boot ROM as part of Boot Loader “blob”, or by Boot Loader). The TSEC FWs (both RNG and Driver FWs) are authenticated by TSEC.

Each Falcon is expected to contain enough internal RAM to support the entirety of its firmware load. If for any reason the required firmware grows beyond the size of this internal RAM, it will be necessary for the firmware image to remain in DRAM so that Falcon may swap in the required portion for a given workload.

12.1.6 Boot Loader

The Boot Loader is read and invoked by Boot ROM or MicroBoot from the boot media, and executes initially on the AVP and primarily on the CPU. It is significantly larger in size than the MicroBoot, and the boot media should ideally be configured to its highest speed before the Boot Loader is read. The Boot Loader is generally the responsibility of the OEM, although NVIDIA may provide a reference implementation and/or substantial support. The Boot Loader is responsible for remaining system configuration, may display a splash screen or include a system update mechanism, and will eventually hand off to OS boot. There are several different Boot-Loader implementations.

- ***Boot Loader***: NVIDIA developed boot-loader used by Android/Linux
- ***UEFI***: Public domain APIs based NVIDIA implemented Boot Loader, used by WoA.
- ***QuickBoot***: Relatively smaller and faster (than Boot Loader), used by Automotive.

- ***U-Boot***: open source Boot Loader used by many SoCs / OEMs (e.g., used by Google Chrome).

Boot Loader Implementation

NVIDIA recommends that you do NOT program the FUSE_ARM_DEBUG_DIS fuse. Because DBGEN, NIDEN, SPNIDEN, and SPIDEN are all fixed at zero, they disable all access to CoreSight debug and performance counters.

The proper recommended usage is:

1. Leave FUSE_ARM_DEBUG_DIS unprogrammed (disabled, value = 0)
2. Use the SecureJtagControl bit in the BCT to communicate the security level to the Boot ROM/Boot Loader. The Boot Loader then takes the appropriate action to program the DBGEN.
3. The Boot Loader can be used to enable/disable the NIDEN, SPIDEN, and SPNIDEN bits.

> **Note:** When the SecureJtagControl bit in the BCT is either not provided or 0, then NIDEN remains at 1 while DBGEN, SPIDEN, and SPNIDEN remain at 0.

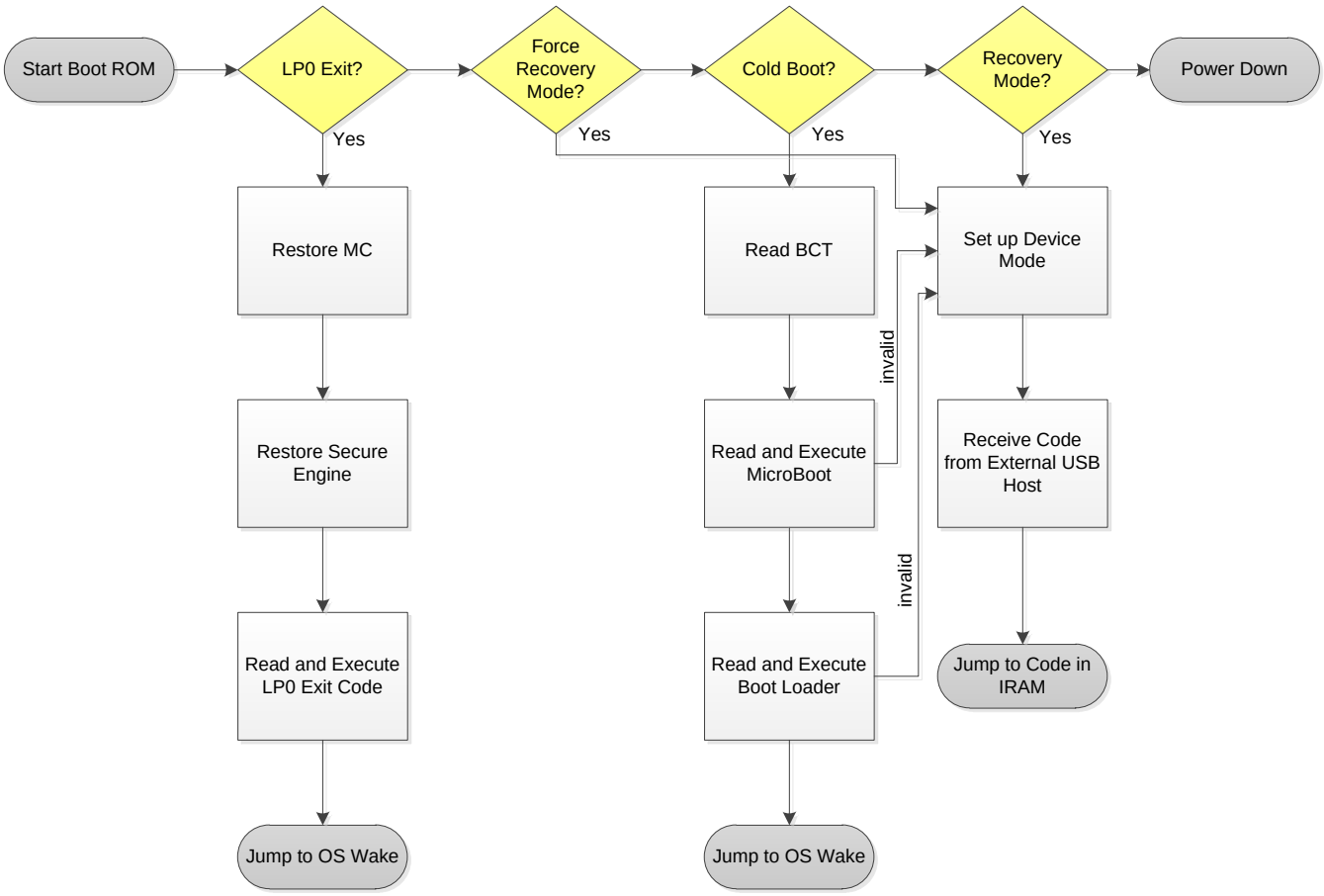
12.1.7 LP0 Exit (LPX) Code

During LP0 exit, Boot ROM configures MC/EMC (by restoring their context from PMC scratch registers) and authenticates LP0 restore code. After that, the Boot ROM hands the control to LPX which was loaded into DRAM (by the Boot Loader) during initial cold boot. The LPX executes on the AVP, communicates with the outside world solely via the PMC scratch registers, and is responsible for start up of the main CPU and hand off to the OS wake vector.

Since LPX executes on AVP, it cannot access TZ-only registers. Also, it cannot access platform-specific data from DRAM (LPX is location independent code) -- LPX can use PMC-scratch or fuses for platform-specific data.

12.2 Boot Sequences

Figure 33: Boot Sequences Overview



12.2.1 Cold Boot

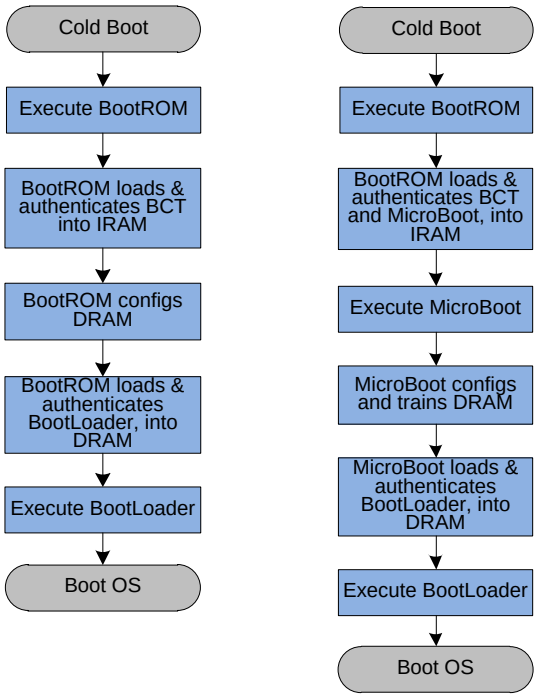
Cold Boot refers to the full boot sequence occurring upon full system reset. Cold boot is normally an uncommon occurrence on tablets and phones, and is not optimized for latency. However, some embedded applications and alternate operating systems will require a faster boot, and may require MicroBoot and other techniques for fast boot.

- [HW] AVP comes out of reset at the Tegra oscillator frequency (12-48MHz), begins executing Boot ROM from IROM.
- [BR AVP IROM] Raise AVP frequency to 216 MHz.
- [BR AVP IRAM] USB Dead Battery Charging - If the battery is dead, and device is connected to a USB Host, stop boot process and charge battery.
- [BR AVP IROM] Configure boot media based on fuses and straps.
- [BR AVP IROM] If PKC authentication, then Read and Validate Public Key from boot media into IRAM. (**Note**: Boot authentication can be PKC (public key) or SBK (symmetric key) based on Fuse. All of the boot components will be authenticated by the same option.)
- [BR AVP IROM] If invalid, go to Recovery Mode.
- [BR AVP IROM] Read and authenticate BCT from boot media into IRAM. (**Note**: After every DMA (from boot media to IRAM), the Boot ROM needs to wait 1 µs before using DMA-ed data)
- [BR AVP IROM] If invalid, go to Recovery Mode.

- [BR AVP IROM] Perform BCT-based system configuration. Optimize boot media performance.
- [BR AVP IROM] Read and authenticate MicroBoot from boot media into IRAM. *****
- [BR AVP IROM] If invalid, go to Recovery Mode.
- [BR AVP IROM] Lock down security. Restrict read access to SE key slots. (**Note**: Even if the Boot ROM goes to Recovery Mode, lock down still happens as the last step of the Boot ROM. This is not a security risk because commands RCM can use are not considered secure risk.)
- [BR AVP IROM] Jump to MicroBoot in IRAM.
- [MB AVP IRAM] Configure DRAM at each supported FV point.
- [MB AVP IRAM] Switch to highest DRAM frequency.******
- [MB AVP IRAM] Configure DRAM
- [BR/MB AVP IRAM] Read and Authenticate Boot Loader and TSEC-RNG from boot media into DRAM.
- [BR/MB AVP IRAM] If invalid, go to Recovery Mode.
- [BR/MB AVP IRAM] Jump to Boot Loader initial AVP code in DRAM.
- [BL AVP IRAM] Trigger TSEC to generate random numbers for keys
- [BL AVP DRAM] Initialize and bring bootstrap CPU core out of reset. Switch to executing the Boot Loader on main CPU.
- [BL CPU DRAM] OEM-specific platform initialization.
- [BL CPU DRAM] Display Splash Screen
- [BL CPU DRAM] Load and authenticate LP0 Exit Code.
- [BL CPU DRAM] Read and optionally authenticate OS boot sector.
- [BL CPU DRAM] Lock any Boot Loader-privileged security / sticky bits.
- [BL CPU DRAM] Hand off to OS.
- [OS CPU DRAM] Boot OS.
- [OS CPU DRAM] MSENC FW, and TSEC FW.

Note:	The MicroBoot layer is optional in Tegra 4 processors. If MicroBoot is not used, then steps listed under MicroBoot can be done by the Boot Loader.

Figure 34: Cold Boot Flow (without and with MicroBoot)



12.2.2 LP0 Exit

In most design applications except some embedded applications, Tegra 4 processors will spend the most time in LP0. LP0 Exit will therefore be the most common form of reset. Exit latency is critical to overall system responsiveness. There LP0 Exit process must not depend on things such as access to the boot media, or soft RSA for LP0 restore code authentication. Any required information must be stored in the PMC scratch registers or in DRAM itself, and hardware RSA should be used for LP0 restore code authentication.

- [HW] AVP comes out of reset at osc (normally 12 MHz), begins executing the Boot ROM from IROM.
- [BR AVP IROM] Raise AVP frequency to 216 MHz.
- [BR AVP IROM] Detect LP0 Exit.
- [BR AVP IROM] Restore “safe” memory configuration from PMC registers.
- [BR AVP IROM] Restore optimized memory configuration from DRAM.*****
- [BR AVP IROM] Initiate SE restore.
- [HW] Decrypt and install SE state from secure PMC registers.
- [BR AVP IROM] Authenticate LP0 Exit Code.
- [LX AVP DRAM] Initialize bootstrap CPU core. Start executing LP0 Exit Code on main CPU.
- [LX CPU DRAM] OEM-specific platform restore.
- [LX CPU DRAM] Hand off to OS wake vector.

Note: * This step may be required if DRAM training is employed

12.2.3 Recovery Mode

When the boot media is blank or corrupted, Recovery Mode (RCM) can be used to inject code into IRAM to begin a re-installation process. In Recovery Mode, the USB controller is initialized into device mode. Recovery code provided by an external USB host running NVIDIA software will be loaded into and run from IRAM. Recovery Mode is the mechanism by which boot components are initially signed and installed to the boot media during manufacturing.

- [BR AVP IROM] RCM initiated during cold boot by
  - Detect hardware strap, or
  - Detect invalid boot assets, or
  - Detect designated PMC scratch bit set.
- [BR AVP IROM] Initialize USB into device mode. (Note: only USB2 (also known as Synopsys USB IP) can be used for recovery)
- [BR AVP IROM] Receive code from external USB host into IRAM.
- [BR AVP IROM] Check integrity and optionally authenticate recovery code.
- [BR AVP IROM] If invalid, power down.
- [BR AVP IROM] Execute recovery code from IRAM.

12.3 Functional Requirement

12.3.1 Boot Security

The Boot ROM is the root-of-trust for the boot process. The Boot ROM is effectively hard-wired onto the chip, and is always the first piece of code to run upon reset. Each boot component has the ability to carry forward the chain-of-trust by authenticating and optionally decrypting the next boot component.

Sensitive information in the system including the readability of AES keys are cleared or locked via sticky bits before exiting the Boot ROM. Note that the AES keys, even when readability is disabled, can remain present within designated key slots of the SE so that they may be used to authenticate/decrypt later boot components.

In order to grant certain privileges to the Boot Loader, some access items may be locked down later by the Boot Loader before handoff to the OS. One example is that locking writability of the media in a protected boot design may be done by the Boot Loader so that it can provide system upgrade/recovery services if needed.

In addition to the hardware AES (symmetric key cryptography) engine located within the SE. The SE adds hardware RSA (public/private key cryptography – also known as PKC) implementation. The hardware RSA adds hundreds of microseconds of latency per signature calculation.

Tegra 4 processors can use either AES or RSA boot authentication based on a fuse setting.

12.3.1.1 AES Secure Boot

In case AES is used:

- Security depends upon keeping the key(s) hidden from OS/application software,
- Security is improved by employing as many different keys as possible; ideally a unique key per individual system, and
- Existing signing tools and manufacturing flow support this model.

Tegra 4 devices use 128-bit SBK (Secure Boot Key) fuses to store per system unique AES keys. The SBK fuses need to be programmed in a secure facility, which makes SBK boot authentication problematic for some customers. An alternative is to use RSA authentication.

The boot code can be optionally encrypted. The encryption/decryption is also done using same SBK keys.

12.3.1.2 RSA (also known as PKC) Secure Boot

In case RSA is used:

- The public key may remain present in memory; hiding it is not necessary,
- Security depends upon the OEM keeping their private key a secret, and
- Signing tools and manufacturing flow will need to change significantly.

The 2048-bit of public key is stored in the flash device. The 256-bit SHA hash of public key is stored in on-die fuses (PKC fuses) to validate public key from the flash device. The private key remains with OEM. Therefore, we cannot sign any generated code/information (which would require a private key) on-the-fly using PKC. As a result, any update to BCT, MicroBoot, or Boot Loader must be signed ahead of time by OEM (or OEM server for OTF/OTA updates). Anything which cannot be signed ahead of time will have to be either unsigned (if not a security risk, e.g., NAND bad-block-table), or handled differently (e.g., NAND start-sector-address which was generated as part of Boot Loader-information).

With RSA secure boot, LP0 restore code will be signed (as a separate component) using PKC. And at the LP0 exit, LP0 restore code will be authenticated using PKC.

With RSA secure boot, encryption/decryption (if required) would still be done using SBK keys.

12.3.2 USB

12.3.2.1 USB Dead Battery Charging

When the battery is dead or too weak to support a system boot, then it may be necessary to use power from a USB host or charger to boot the system and/or recharge the battery.

Tegra systems with a dead battery (and not connected to an AC power adapter) are may be able to boot when connected to a USB host, if they can operate within its power restrictions. Prior implementations assumed that the USB VBUS can provide up to 500 mA. This is not always in compliance with the USB specification. To comply with the specification, we can only assume that USB VBUS can provide up to 100mA, and only after negotiating with the host can the Tegra 4 system draw more than 100mA current from the USB VBUS.

Tegra 4 devices provide support for USB dead battery charging specification compliance. The compliance support is enabled by a fuse (which is default to disabled).

If compliance is enabled, then using 100mA is not enough to power LCD backlight, mass storage, DRAM, etc. Therefore USB charging support is implemented in the early Boot ROM, which provides support for negotiating with the charger (but does not provide support to negotiate with the host for being enumerated as high current *device*). The Boot ROM starts with the assumption that the USB VBUS can always provide 100mA current. It negotiates with the charger for higher current. If the host can provide >500mA current then boot continues; otherwise boot stalls and remains stalled for as long as dead/weak battery condition persists.

If the OEM wishes to have a different dead battery charging solution, they can have the Boot ROM based USB dead battery charging solution disabled (by not setting fuse) and implement their own solution in the MicroBoot. This is a possible option if the OEM platform is capable of powering the boot media (which is required for loading MicroBoot) while remaining under a 100mA power budget.

12.3.3 Watchdog Timer Usage

The Watchdog Timer (WDT) is intended to cover the situations where the system hangs in the boot phase for unexpected reasons. It may be due to flash device problem, power noise, or signal noise margin, or soft error, or some other error condition. The Watchdog Timer forces a reboot in these circumstances.

To be able to detect such hangs and reboot the system, an internal watchdog timer can be used. Tegra 4 devices provide five WDTs (one each for CPU0/1/2/3 and one for AVP) which can be configured to detect such hangs. Any one of these WDTs can be configured to generate a WDT interrupt (at first expiry of the timer) and/or reset the system (at fourth expiry of the timer).

If the WDT-enable fuse is set, then (at both cold and warm boot), the Boot ROM enables WDT0 as-early-as possible in the Boot ROM boot phase. If the WDT expires, then the PMC resets the *chip* similar to a thermal shutdown reset. As a result of this reset, the system reboots. During the reboot, the Boot ROM reads the cause of reset. If it was a WDT caused reboot and the PMIC command has been stored in PMC scratch, then the Boot ROM performs a *system* reset via the PMIC.

At cold boot, when the Boot ROM transfers control to the Boot Loader (or MicroBoot), then it leaves WDT4 enabled. The Boot Loader is expected to keep WDT4 enabled all through the Boot Loader (at AVP) phase, halt WDT4 when AVP is halted, and start WDT0 when the Boot Loader on CPU0 takes control. The Boot Loader will disable WDT0 just before transferring control to OS. Similarly, at warm boot (LP0 exit), when the Boot ROM transfers control to LP0 restore, then it leaves WDT4-enabled. LP0 restore is expected to keep WDT4 (for AVP)/WDT0 (for CPU0) enabled all through restore phase, and disable it just before transferring control to OS.

12.3.4 Thermal Shutdown Requirement

If the chip temperature is so high that it can cause permanent physical damage or chip cannot reliably function, then it is necessary to "shut down" the chip. For this purpose, soc_therm unit provides a mechanism which (if enabled) detects thermal shutdown condition (THERMTRIP) and asserts a signal to PMC to request thermal shutdown.

At the assertion of THERMTRIP condition, PMC hardware (if enabled) resets the *chip*, but stores the cause of reset (in APBDEV_PMC_RST_STATUS register) and retains contents of scratch registers. As a result of this reset, system reboots (in a similar manner to a cold boot). During this reboot, the Boot ROM reads the cause of reset, and if it was THERMTRIP caused reboot then the Boot ROM performs *system* reset or system shutdown via PMIC or GPIO. To avoid PMIC specific code in the Boot ROM, Boot Loader (at cold boot) stores the PMIC command (for system reset or shutdown) in PMC scratch, and the Boot ROM simply sends that command to PMIC.

THERMTRIP Detection Enable

To enable THERMTRIP, soc_therm has to be configured with sensor calibration values (from fuses) and THERMTRIP threshold (usually from boot media). Software will configure soc_therm to enable THERMTRIP detection, enables PMC thermal reset option, and stores THERMTRIP threshold in the PMC scratch register.

At LP0 exit, software will reconfigure soc_therm calibration values from fuses (or DRAM), and THERMTRIP threshold from PMC scratch (or DRAM depending upon whether LP0 restore of Kernel restore restores soc_therm).

12.3.5 CPU Cold Boot

At cold boot, the Boot Loader (on AVP) determines which CPU cluster (fast or shadow) to boot from. At boot time, the OS needs to discover the maximum number of CPUs, and allocate resource accordingly. So for this reason Tegra boots with the fast cluster (cluster0) to be able to discover maximum number of CPUs. An optimization can be to start boot from shadow cluster (cluster1), for example in the case of not enough battery power, or very high temperature.

At power-on-reset, all of the CPUs (fast and shadow) are power gated. In addition, shadow L2 is power gated, and CPU rail is off. Refer to PMC_PWRGATE_STATUS register description for the default of CPUs power status.

The Boot Loader powers up the CPU rail by configuring the PMIC (via I2C). At this stage CPU rail's status in APBDEV_PMC_PWRGATE_STATUS[CRAIL] would still be power-off (since the CPU rail is powered up without PMC hardware involvement). The Boot Loader needs to ensure that APBDEV_PMC_PWRGATE_STATUS[CRAIL] gets updated to power-on. It can be done as follows.

1. Power up the CPU rail by configuring PMIC via I2C
2. Configure PMC's CPU power good timer

Note if this configuration is done before next step (step #3), then the Boot Loader does not have to wait for CPU rail power good (instead it can just wait for PMC_PWRGATE_STATUS[CRAIL] status to be power on – step #4, which would be more efficient).

3. Request PMC to power up CPU rail (by writing PMC_PWRGATE_TOGGLE)

   At this stage cpu_pwr_req pad is disabled, so this power up request just updates the PMC status
4. Wait for PMC_PWRGATE_STATUS[CRAIL] status to be powered on
5. Enable cpu_pwr_req (by writing PMC_CNTRL)

After CPU rail power up, the Boot Loader power ungates CPU0 which can be done as follows.

1. Power ungate CPU0 (by PMC_PWRGATE_TOGGLE register write)
2. Removes CPU rail clamp (by PMC_REMOVE_CLAMP_CMD register write)
3. Enables CPU clock (by CAR register write)
4. Deassert C0NC reset followed by CPU0 reset (by CAR register write)

   Immediately after reset deassertion, CPU0 will start fetching from reset vector

Note, if the hardware default of CPU reset vector needs to be updated, then the Boot Loader (on AVP) will have to configure CPU reset vector (in EVP_CPU_RESET_VECTOR register) before transferring control to the Boot Loader (on CPU0).

After transferring control to CPU0, the Boot Loader (on AVP) puts AVP into halt. The Boot Loader (on CPU0) boots the CPU0, and then transfers control to OS. The CPUs other than CPU0 are powered up by OS.

12.3.6 IRAM Requirements

The Tegra 4 processor boot process currently requires 256KB of IRAM space.

There are two main requirements for IRAM space, both of which are during cold boot and largely depend upon MicroBoot size. One is during the Boot ROM phase, where the MicroBoot binary image must be resident, and the other is during MicroBoot execution phase, where both the code and data footprint of MicroBoot must be considered.

These are the current footprint sizes:

	Tegra 4 Estimate (KB)
Boot ROM phase footprint	216
MicroBoot phase footprint	170

12.3.7 Boot Devices

Tegra 4 series processor supports boot devices listed in the following table. Boot device frequency/mode requirements are also listed in the following table.

Table 41: Boot Devices Frequency/Mode Targets

Device	BR Frequency	MB/BL Frequency	Notes
eMMC	18MHz SDR (PPLP/DivN)	52MHz DDR 200MHz SDR	Tegra 4 devices use @52MHz DDR (=104Mt/s) 200 MHz SDR.
SPI	OSC/2	Device limited (<50MHz)	The Boot ROM does not support the SPI FAST_READ command, which is normally required for SPI > 20 MHz. Thus SPI > 48 MHz of OSC frequency is not supported unless the SPI device works without FAST_READ.

12.4 Performance Requirements

The following are boot performance targets followed by reference measurements from previous chips. These targets assume the AVP coming out of reset at 216 MHz.

12.4.1 Cold Boot Latency

TBD

12.4.2 LP0 Exit Latency

Tegra 4 LP0 Exit latency target:

- 5 ms from wake event to OS wake vector.

12.5 TSense Reset Handler

The TSense reset is handled at the beginning of the Boot ROM secure ROM entry. The PLLP and PMC clocks are available for this reset.

The SCRATCH54 and SCRATCH55 registers deal with PMIC board-specific support (these are defined in the Power Management Controller section). The PMIC driver configures them. The I2C controller is used to communicate to the PMIC.

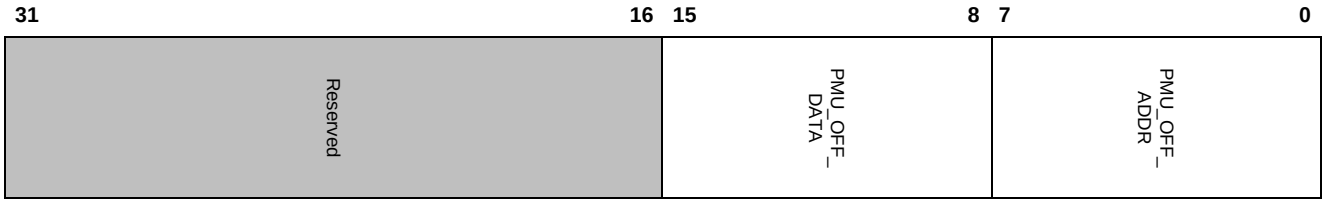
12.5.1 Scratch Registers

The SCRATCH54 and SCRATCH55 registers are used in the TSense reset handler in I2C mode.

Note: The Reserved bits must be set to 0.

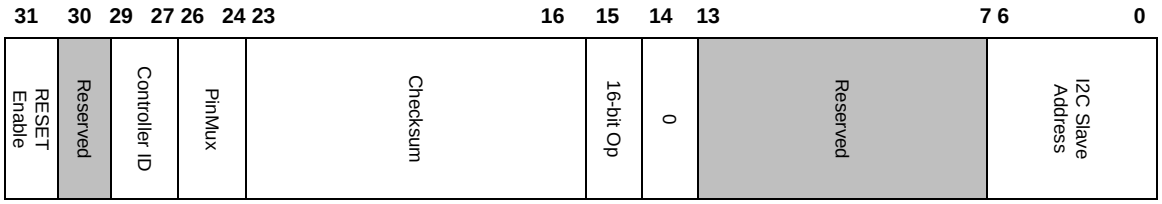
SCRATCH54 (PMIC Scratch Register A)

In I2C mode, the SCRATCH54 register bits are configured as shown below.



12.5.1.1 SCRATCH55 (PMIC Scratch Register B)

In I2C mode, the SCRATCH55 register bits are configured as shown below.



12.5.2 Pinmux Support

All Standard I2C pinmux signals are supported as shown for the three configurations below. Mixed pinmux configurations are not supported; for example, GEN1_I2C_SCL -> I2C1_CLK, SPDIF_IN -> I2C1_DAT is not supported. The I2C peripheral clock is targeted at 100 kHz (Standard mode). The Controller ID and PinMux columns in the following table are from the SCRATCH55 register.

I2C Controller	Controller ID	I2C Signals	Config 1	Config 2	Config 3
			PinMux = 0	PinMux = 1	PinMux = 2
I2C_1	0	I2C1_CLK	GEN1_I2C_SCL	SPDIF_OUT	SPI2_CS1_N
		I2C1_DAT	GEN1_I2C_SDA	SPDIF_IN	SPI2_CS2_N
I2C_2	1	I2C2_CLK	GEN2_I2C_SCL	-	-
		I2C2_DAT	GEN2_I2C_SDA	-	-
I2C_3	2	I2C3_CLK	SDMMC4_CMD	CAM_I2C_SCL	-
		I2C3_DAT	SDMMC4_DAT4	CAM_I2C_SDA	-
I2C_4	3	I2C4_CLK	DDC_SCL	-	-
		I2C4_DAT	DDC_SDA	-	-
I2C_PMU	4	I2CPMU_CLK	PWR_I2C_SCL	-	-
		I2CPMU_DAT	PWR_I2C_SDA	-	-

12.5.3 Checksum Calculation

Every byte in the SCRATCH54 and SCRATCH55 registers must add to 0. The two scratch registers contain 8 bytes total.

To calculate the checksum:

- Write 0 to the 8-bit checksum byte
- Add bytes 0 through 7 where: sum = ~sum+1 (2’s complement of sum)
- Write sum to the 8-bit checksum byte.

12.6 Boot ROM Fuses and Straps

Table 42: Boot ROM Related Fuses

Number of Bits	Location	Function	Values / Notes
1	FUSE_SECURITY_MODE	OdmMode and JTAG disable	Boolean
1	FUSE_PRODUCTION_MODE	ProductionMode	Boolean
1	FUSE_FA	Failure Analysis Fuse Mode Set	Not-zero = FA
6	FUSE_OPT_FAB_CODE[5:0]	Fab code	
32	FUSE_OPT_LOT_CODE[31:0]	Lot code	
4	FUSE_OPT_VENDOR_CODE[3:0]	Vendor code	
6	FUSE_OPT_WAFER_ID[5:0]	Wafer ID	
9	FUSE_OPT_X_COORDINATE[8:0]	X coordinate	
9	FUSE_OPT_Y_COORDINATE[8:0]	Y coordinate	
128	FUSE_PRIVATE_KEY0 FUSE_PRIVATE_KEY1 FUSE_PRIVATE_KEY2 FUSE_PRIVATE_KEY3	Secure Boot Key (SBK)	128-bit AES key
32	FUSE_PRIVATE_KEY4	Device Key (DK)	32-bit key
14	FUSE_BOOT_DEVICE_INFO[13:0]	Boot Device Config	See below
3	FUSE_RESERVED_SW[2:0]	Boot Device Select	0x0 = eMMC 0x1 = Reserved 0x2 = SPI 0x3 = Reserved 0x4 = Reserved 0x5 = Reserved 0x6 = Reserved 0x7 = Reserved
1	FUSE_RESERVED_SW[3]	Skip Device Selection straps	Boolean
8	FUSE_SKU_INFO[7:0]	SKU	Only [3:0] used in the Boot ROM
	FUSE_RESERVED_SW[7:6]	SW Reserved	
	FUSE_RESERVED_SW[5]	ENABLE_WATCHDOG	ENABLE_WATCHDOG = 1 enables watchdog. This fuse should be set to 1 for ODM mode only
3	FUSE_RESERVED_SW[4]	SW Reserved	ENABLE_CHARGER_DETECT
256	FUSE_PUBLIC_KEY0 FUSE_PUBLIC_KEY1 FUSE_PUBLIC_KEY2 FUSE_PUBLIC_KEY3 FUSE_PUBLIC_KEY4 FUSE_PUBLIC_KEY5 FUSE_PUBLIC_KEY6 FUSE_PUBLIC_KEY7	Public key hash	SHA-256 (256 bits) hash of public key modulus
1	FUSE_PKC_DISABLE	PKC disable control	Not-zero = PKC disable

Table 43: Fuse Boot Device Information

Device	Fuse Bits	Description	Values
eMMC	13	Reserved	Ignored; set to 0x0
	12:10	MultiPage support	0x0 = default Single page read (512 Byte)
			0x1 = Multi 2 page read (1024 Byte)
			0x2 = Multi 4 page read (2048 Byte)
			0x3 = Multi 8 page read (4096 Byte)
			0x4 = Multi 16 page read (8192 Byte)
			0x5 = Multi 32 page read 16384 Byte)
	9:6	Clock Divider PLLP clock @ 216MHz	0x0 = default clock divider 11 (clock @ 19.63 MHz)
			0x1 = clock divider 10 (clock @ 21.6 MHz)
			0x2 = clock divider 9 (clock @ 24 MHz)
			0x3 = clock divider 8 (clock @ 27 MHz)
			0x4 = clock divider 7 (clock @ 30.8 MHz)
			0x5 = clock divider 6 (clock @ 36 MHz)
			0x6 = clock divider 5 (clock @ 43.5 MHz)
			0x7 = clock divider 4 (clock @ 27 MHz)
			0x8 = clock divider 3 (clock @ 36 MHz)
			0x9 = clock divider 2 (clock @ 27 MHz)
	5	VDDIO_SDMMC4 Pads voltage	0x0 = 1.8V (power on default)
			0x1 = 1.2V
	4	Disable Boot Mode	0x0 = Boot mode On
			0x1 = Boot mode Off
	3:2	Voltage Range	0x0 = Query Voltage
			0x1 = High Voltage
			0x2 = Dual Voltage
			0x3 = Low Voltage
	1	DDR Mode Selection	0x0 = Normal
			0x1 = DDR
	0	Data bus width	0x0 = 4 bits
			0x1 = 8 bits
NAND	13:12	Toggle DDR	0x0 = Discovery (valid for <= 42nm NAND)
			0x1 = Disabled
			0x2 = Enabled
			0x3 = Reserved
	11:10	ErrorFree NAND Controller Version	0x0 = Disable Error Free
			0x1 = EF NAND 1.0
			0x2 = EF NAND 2.0
			0x3 = EF NAND 3.0
	9	BCH sector size	0x0 = 512 bytes
			0x1 = 1024 bytes
	7:8	Main ECC offset	0x0 = 4 byte

Device	Fuse Bits	Description	Values
			0x1 = 8 bytes
			0x2 = 12 bytes
			0x3 = 16 bytes
	6	Page/Block size offset	Page/Block size from read id is multiplied by 2^Offset to correct the interpretation of the data (< = 42nm NAND support).
			0x0 = 0
			0x1 = 1
	5	ONFI disable	0x0 = ONFI support on
			0x1 = ONFI support off
	4:3	Data bus width	0x0 = Discovery (valid for > 42nm NAND)
			0x1 = x8
			0x2 = x16
			0x3 = reserved
	2:0	ECC strength	0x0 = Discovery
			0x1 = BCH4
			0x2 = BCH8
			0x3 = BCH16
			0x4 = BCH24
			0x5 = ECC Off
SPI Flash	13:1	Reserved	Ignored; set to 0x0
	0	Page size (Block size is fixed to 32K due to block erase command's limitation)	0x0 = 2K
			0x1= 16K (Boot ROM data buffer size). This will be used for performance optimization.

Table 44: Boot ROM Straps

Number of Bits	Location	Function	Values	Notes
2	RAM_CODE[1:0]	Selects SDRAM configuration set within the BCT	STRAPPING_OPT_A[5:4]	-
2	RAM_CODE[3:2]	Selects secondary boot device configuration set within the BCT	STRAPPING_OPT_A[7:6]	-
1	FORCE_USB_RECOVERY	Forces USB Recovery Mode	STRAPPING_OPT_A[25]	-
4	BOOT_SELECT[3:0]	Device Selection	STRAPPING_OPT_A[29:26]	-
			0x0 = eMMC x4 BootModeOn	Maps to SDMMC with config = 0x0000
			0x1 = eMMC x8 BootModeOn	Maps to SDMMC with config = 0x0001
			0x2 = Reserved	Reserved
			0x3 = NAND	Maps to Legacy NAND with config = 0x1000
			0x4 = NAND with block and page offset = 1 (< 42nm NAND)	Maps to NAND with config = 0x00C8
			0x5 = Reserved	Reserved
			0x6 = Reserved	Reserved

Number of Bits	Location	Function	Values	Notes
			0x7 = EF NAND with block and page offset = 1(< 42nm NAND)	Maps to EF NAND with config = 0x08C8
			0x8 = SPI Flash	Maps to config with page size = 2K
			0x9 = Reserved	Reserved
			0xA = eMMC x4 BootModeOFF	Maps to SDMMC with config = 0x0010
			0xB = eMMC x8 BootModeOFF	Maps to SDMMC with config = 0x0011
			0xC = eMMC x8 BootModeOFF Clk Divider 4	Maps to SDMMC with config = 0x0111
			0xD = SATA	Maps to SATA with config = 0x0000
			0xE = eMMC x8 BootModeOFF Clk Divider 6	Maps to SDMMC with config = 0x0191
			0xF = Use fuse data	Maps to actual fuse values

13.0 HOST SUBSYSTEM

The Host1x unit provides the programming interface to various graphic and video engines, and display. In addition to Host1x's interfaces with these units, it also has an IP interface to the ARM7 crossbar (xbar), TSEC, and a memory interface. Commands are either gathered from a push-buffer in memory or provided directly by the CPU, and then supplied to the clients behind Host1x via Host1x channels. The channels also provide a means of synchronization between software and any individual block or amongst the blocks themselves via hardware Sync Point signals (syncpts).

13.1 Glossary

- **Channel** – A piece of hardware that provides a programming interface to one or more classes. A channel contains a sequence of commands embodied in a command FIFO. This sequence of commands can also be thought of as a thread of execution and of a single context. There exists only one sequence of commands or context per channel. That is to say, there is no hardware-managed context switching within a single channel.
- **Command FIFO** – Holds channel commands supplied by the CDMA or PIO accesses. It is stalled by synchronization methods, backpressure from its destination client, or ownership of the destination client by a different channel
- **Protected Channel** – Dedicated channel used by software for flow control.
- **Channel Commands** – Entries in the command FIFO. Commands take a common form interpretable by Host and are used for 3 main functions: controlling class ownership and class virtualization of the channel; command expansion via gather commands; and issuing class methods.
- **Client, device, resource** – A piece of hardware that resides behind Host, connected via the HRD and HRW buses. Client is the preferred terminology for this document. Generally mapped one-to-one with a class, although this is not a strict requirement.
- **Method** – An operation on a class. The most common example is a single register write.
- **Class method –** A method that belongs to an unspecified class.
- **Host method** – A method that belongs to the Host class.
- **Class** – An abstraction of a client, device, or resource. It is a collection of methods. A class can only be owned by multiple channels, but only one channel may actively be using any class, which is known as a channel's working class. The transfer of working class from one channel to another is known as a context switch, which is managed by Host.
- **Class ID** – A unique tag corresponding to each class.
- **SetClass** – A channel command that takes a class ID as an argument. SetClass is the sole means to indicate the current virtualization of the channel. It also the sole means that a channel can acquire ownership of a class. Subsequent channel commands are directed towards this class.
- **Gather** – A channel command to gather contiguous chunks of memory and inserts it into the command stream.
- **Nonincrement** – A channel command that specifies an offset and a count. The offset specifies a method in the current class and the count indicates the number of subsequent words to be written at that offset.
- **Increment** – A channel command that specifies an offset and a count. Increment works like nonincrement, except the offset is incremented by one on each on each write.
- **Mask** – A channel command that specifies an offset and a mask. A mask command works much like increment and non-increment, except the offsets are calculated by looking at the bits set in the mask. The bit position indicates the relative offset from the specified offset in the command. Mask expects the number of words to follow to be equal to the number of bits set in the mask.
- **Channel switch** – A transfer of ownership of a client from one channel to another; a type of context switch. Sometimes a source of confusion, this is not a transfer of ownership of a channel, but of a client. This is the preferred narrowed nomenclature to context switch in order to avoid confusion.

- **Class switch** – A change in the current in the active class of a client; a type of context switch. In the event that a class switch also involves a channel switch, channel switch is the preferred declaration.
- **Context switch** – A Host event where it facilitates a client's state transition. A context switch is either a channel switch or a class switch.
- **CDMA** – The command DMA. It is responsible for reading commands from memory and supplying them to all channels. It starts from the address in the channel's DMASTART register and continues until it reaches the address in the channel's DMAPUT register.
- **DMAGET** register – Holds the current address of the CDMA. One exists per channel.
- **DMAPUT** register – Holds the end address of a command stream in memory. One exists per channel.
- **DMASTART** register – Holds the start address of a command stream in memory. One exists per channel.
- **Direct register access –** Access to a client initiated by the CPU. If a channel and the CPU initiate a data transaction to the same client at the same time, the CPU gets priority. Direct register accesses are orthogonal to channels in regards to client ownership.
- **Host Master –** General term to indicate an entity that can control Host. The current list is the MPCore™, COP, and TSEC.
- **Push-buffer** – A set of commands residing contiguously in memory. It is a communication method between the processor and Host. Commands are placed at the end of the buffer and a pointer is updated in Host.
- **Indirect register access** – Access to a client initiated by the CPU, but requires two Host register accesses. This is deprecated in the SOC version of the host; it comes from the companion-chip mode where fixed-latency interfaces to the CPU may exist. Like direct register accesses, indirect takes priority over channel transactions. Indirect register accesses must be tied a channel because they require a read return FIFO.
- **WAIT** – A host method that takes a single vector argument. It is used in conjunction with syncpt. A WAIT stalls the command FIFO until the supplied vector intersects the RAISE register at all.
- **Tick Count** – Running count of the Host clock after it has been enabled.
- **Sync Point (Syncpt)** – A counter used for synchronization. Every time a specified event occurs, the counter is incremented. A channel can wait until the syncpt attains a specified value, or an interrupt can be generated when a specified value is reached.
- **Teardown** – Can be either a module or channel teardown. Essentially, it means all links and references within the Host between the specified module or the specified channel are removed and reset.

13.2 Features

13.2.1 Class Based Programming

Desktop GPUs support a programming interface that is based on writing to offsets within a "class" that implements a function, rather than to specific register offsets and formats.

By using "class" interface register offsets/formats need not to remain fixed so there is nothing chip specific in the API. This will allow both hardware flexibility and software driver compatibility.

13.2.2 Command Buffer DMA

To maximize the Host1x bandwidth, it is important to burst as many write cycles as possible, which requires processor to first combine writes. The write buffering typically requires sequential offsets to be written and also may not follow strict programming order. The CPU will first store this buffered data directly to memory and then program Host1x engine to DMA it.

13.2.3 Multiple Channels

A channel is a thread of execution within Host1x. Similar to a multithreaded CPU, a channel helps defining a context that can be used to allow multiple users of the Host1x and plays a role in context switching.

A channel switch does not always require a client module context switch. There will be a state change within the Host1x, but if channels have non-overlapping usage (e.g., possibly a 3D channel and MSENC encode channel), there may be no need to context switch any Host1x client module.

13.3 Hardware Features

13.3.1 Channels

The Tegra® 4 series processor has nine Host1x channels. Each channel has a set of registers and a command FIFO. Each channel can be associated with one or more Host1x clients. The channel is the primary means of delivering commands to clients, which is described in the Host1x Programming Model section.

13.3.1.1 Channel Registers

Channel registers are broken into two functional groups: one grouping is associated with the command FIFO and command delivery to clients; the other grouping is associated with register and memory access.

CDMA Registers

- **HOST1X_CHANNEL_DMASTART_0:** This register triggers a DMA fetch from memory for this channel, if PUT register does not equal the GET register.
- **HOST1X_CHANNEL_DMAPUT_0:** This register triggers a DMA fetch from memory for this channel, if the PUT register does not equal the GET register. This address is relative to the DMASTART base address. It does not support byte writes. All 4-byte data need to be programmed.
- **HOST1X_CHANNEL_DMAGET_0:** This register tracks the MC offset, which DMA engine has read. It gets incremented as entries are loaded from the channels command buffer into the FIFO. This address is relative to the DMASTART base address.
- **HOST1X_CHANNEL_DMAEND_0:** This is the boundary of illegal addresses (either end of push-buffer or end of physical memory). This is designed to prevent DMA from prefetching illegal addresses. If DMA reaches this address before seeing a RESTART, it will stop. This would be a software error condition.
- **HOST1X_CHANNEL_DMACTRL_0:** The various fields of DMA control register are described as below:
  - **DMAGETRST**: Reset GET pointer to '0'. Useful for cleaning up crashed channels. DMAGET value is not updated instantly. It takes 4 cycles between programming of reset and valid DMAGET.
  - **DMAGETINIT**: Reset GET pointer to the value of DMAPUT when DMAGETRST is asserted.
  - **DMASTOP**: Stop DMA from fetching on this channel.

> **Note:** A Command DMA channel needs to be enabled for PIO-gather to work.

Access Registers

- **HOST1X_CHANNEL_INDOFF_0 & HOST1X_CHANNEL_INDOFF2_0:** The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET value is increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA. The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use

INDOFF2 to set the offset while still using INDOFF to set other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases.

- For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.
- **HOST1X_CHANNEL_INDCNT_0:** Indirect register access count used to trigger indirect reads. Holds the number of registers/memory locations that will be read out. Channels should not request more than there is space available in their output FIFO. Only the protected channel should make liberal use of this feature for speeding up context switching. For indirect frame buffer reads, each channel cannot issue more than NV_HOST1X_MAX_IND_FB_READS at once. The read data must return and be written into the per-channel output FIFO before any additional reads can be issued.
- **HOST1X_CHANNEL_INDDATA_0:** This register, when written, writes to the data to the INDOFFSET in INDOFF. For reads, a REGFNUMEMPTY number of 32-bit values can be read before needing to poll FIFOSTAT again. The per-channel output FIFO (OUTFENTRIES) is readable via this offset. A read of INDDATA will pop an entry off of the per-channel output FIFO.
- **HOST1X_CHANNEL_CMDSWAP_0:** Command swap control. Affects swapping on writes to the PIO region and the frame-buffer buffered memory write region.
- **HOST1X_CHANNEL_FIFOSTAT_0:** CFNUMEMPTY is the number of free slots available in the per-channel command FIFO (needed for PIO or polling for completion of a wait).

13.3.1.2 Channel Command FIFO

The command FIFO is loaded either the CDMA or via PIO access. PIO is mainly used for debug purposes, but it allows a Host1x master to fill the command FIFO with channel commands by writing into “command FIFO” region of the channel's address space. PIO Command FIFO accesses should not be issued while a “Gather” process is busy.

A command FIFO has a set of registers that point to a location in memory where a sequence of channel commands resides. This sequence of commands is generally referred to as a “Push-buffer”.

A Host1x master can write either to DMAPUT or DMASTART register to trigger command fetching by the CDMA. DMASTART indicates where to start fetching, and DMAPUT indicates where to stop. DMAGET is a read-only reference to the present location of the command FIFO; it is incremented when the command is popped from the command FIFO. DMAPUT and DMAGET are relative addresses to DMASTART.

DMAEND provides an upper boundary to prevent the CDMA from fetching illegal addresses; fetching will cease when DMAGET equals DMAEND.

The command FIFO can be halted by writing the DMASTOP field in the Dmactrl register. Dmactrl also provides a mechanism to reset the DMAGET pointer (*Dmagetrst* field) to either 0 or to the value of DMAPUT (Dmainitget field).

13.3.1.3 Channel Commands

Channel commands can be split into two categories:-

- **Class commands:** A class command is a mechanism to communicate a class write to a client. The channel must have an active class when processing class commands. The active class is set by a SETCL (SetClass) command.
- **DMA commands:** DMA commands control what is being fetched. They are processed at the top of the command FIFO while class commands are processed at the bottom of the command FIFO. This is because DMA commands change the command stream and must be processed before entering the FIFO.

Offsets in class commands are relative to the active class' base as they are limited to only the methods in the active class.

Channel Commands

- **SETCL (SetClass)** – A channel command that takes a class ID as an argument. SetClass is the sole means to indicate the current virtualization of the channel. It also means that a channel can acquire ownership of a class. Subsequent channel commands are directed towards this class.
- **NONINCR (NonIncrement) –** Takes an offset and count as arguments. There will be <count> datum following this class command that will be written to the specified offset. Nonincrement indicates that the offset will not be incremented per datum write.
- **INCR (Increment) –** Takes an offset and count as arguments. There will be <count> datum following this command that will be written starting at the specified offset. The offset is incremented after each write.
- **MASK (Mask) –** Takes an offset and count as arguments. The number of datum to write after the command is equal to the number of set bits in the count. Each datum is written to the specified offset plus the next active bit location. For example, a count equal to 0x5, would have 2 datum that are written to (offset+0) and (offset+2).
- **IMM (Immediate) –** Takes an offset and 16-bit datum as arguments. The 16-bit datum is written to the offset.

DMA Commands

- **RESTART (Restart or Jump) –** This command specifies an offset relative to DMASTAT. The next command fetched will be from (DMASTART + offset).
- **GATHER** – Command comprises 2 words and has arguments: offset, insert, type, count, and address. When GATHER is processed, <count> data is fetched from the address and inserted into the command stream. If the <insert> argument is enabled, it will insert either an INCR or NONINCR opcode preceding the fetched data, which is specified by <type>.

DMA commands do not need an active class, but often are processed when an active class exists. In the case of GATHER, if <insert> is enabled, there must be an active class.

13.3.1.4 Channel Control

Channel status and control resides in the synchronous register space of the Host1x address map. These registers are aliased in each channel's space. The details of these registers are as below:-

- **CH<0..8>_STATUS :** Status includes client ownership and whether the channel is blocked or not.
- **CH_TEARDOWN:** Using this register each channel can be reset, which is referred to as a teardown. This means all channel state is cleared and all client and class ownership is relinquished.
- **MOD_TEARDOWN:** Similar to channel teardown, there exists a mechanism to reset modules. Setting this register can clears all state associated with a given client/module.
- **<client>_STATUS** : This register indicates the client's currently active class.

13.3.2 Host1x Class

Host1x has its own class that can be executed from the command FIFO. It is comprised of methods to access client registers as well as methods to control channel flow.

13.3.2.1 INDOFF

The following Host1x methods operate identically to the channel registers used for indirect access. Refer to the Host Channel Registers and Indirect Register Access sections.

- INDOFF
- INDOFF2
- INDCTRL

- INDDATA

INDCNT is absent from the list above. In Host1x master-initiated indirect register reads, INDCNT is written to trigger the read. Conversely, reads from the command FIFO are trigged by a write to INDDATA. Reads issued from the command FIFO are returned to the channel's register return FIFO (return_FIFO).

13.3.2.2 DELAY_USEC

Delay_Usec stalls the command FIFO for the number of microseconds specified. This command has no impact on indirect register accesses if **not** initiated from the command FIFO. The microsecond period is calculated based on setting in the Usec_Clk register. The Usec_Clk register is programmed on the basis of Host1x clock for example if host clock is 250 MHz, then this register should be programmed to a value of 250.

13.3.2.3 TICKCOUNT

Tickcount_Hi, Tickcount_Lo, and Tickctrl can also be controlled from the command FIFO. Their operation is identical whether controlled from the command FIFO or a Host1x master.

13.3.2.4 INCR_SYNCPT

All Host1x modules, including the Host1x itself, implement the Incr_Syncpt class.

Incr_Syncpt <Cond> <Indx>

For the Host1x, this method immediately increments Syncpt[Indx] irrespective of the cond.

13.3.2.5 WAIT_SYNCPT

Wait on syncpt – the command dispatch will stall until the syncpt counter pointed by the index field reaches the threshold value specified in the Thresh field:-

Wait_Syncpt <Indx> <Thresh>

The channel will wait until the following is true:

Syncpt[Indx][15:0] >= Thresh[15:0]

where the “>=” takes into account wrapping (see the “Sync Points (SYNCPTs)” section for more information on wrapping). More specifically, the channel will wait until:

((Syncpt[indx] – thresh) & (1 << 15)) != 0

13.3.2.6 WAIT_SYNCPT_BASE

This method uses Syncpt Base registers to calculate threshold value for channel wait operation. The syncpt index, base index and also optional offset will be a part of Wait_Syncpt_Base command:-

Wait_Syncpt_Base <Indx> <Base_Indx> <Offset>

The channel will wait until the following is true:

Syncpt[Indx][15:0] >= (Syncpt_Base[Base_Indx] + Offset)[15:0]

Where the “>=” takes into account wrapping. See the “Sync Points (SYNCPTs)” section for more information on wrapping.

13.3.2.7 WAIT_SYNCPT_INCR

Wait until syncpt increments:

Wait_Syncpt_Incr <Indx>

The channel will stall until Syncpt[Indx] is incremented. Note that this wait method is subject to race conditions and should be used only for bug work-around.

13.3.2.8 LOAD_SYNCPT_BASE

Load a new value into the Syncpt_Base register:

load_syncpt_base <base_indx> <value>

Syncpt_Base[Base_Indx] will be loaded with <Value>.

13.3.2.9 INCR_SYNCPT_BASE

Add an offset to the value in the Syncpt_Base register:

Incr_Syncpt_Base <Base_Indx> <Offset>

The following operation will be done:

Syncpt_Base[Base_Indx] += Offset

Following are the new methods added for 32-bit sync point comparison:-

13.3.2.10 LOAD_SYNCPT_PAYLOAD_32 <Payload(32)>

This method loads a 32-bit value into the corresponding channel's Channel_Syncpt_Payload register:-

Channel_Syncpt_Payload[31:0]= <Payload(32)>

13.3.2.11 WAIT_SYNCPT_32 <Indx(8)>

This method stalls the current channel until following condition is true:-

(SYNCPT [<indx>][31:0] - PAYLOAD[31:0]) & 0x80000000) == 0

Here the Payload value is taken from Channel_Syncpt_Payload of the current channel.

This is essentially a wrapping stall until ((SYNCPT [<indx>][31:0] >= PAYLOAD[31:0])

13.3.2.12 WAIT_SYNCPT_BASE_32 <Indx(8)> <Base_Indx(8)> :

This method stalls the current channel until following condition is true:-

((Syncpt [<Indx>][31:0] - (Payload[31:0] + Syncpt_Base[<Base_Indx>][31:0]) & 0x80000000) == 0

Here the Payload value is taken from Channel_Syncpt_Payload register of the current channel.

This is essentially a wrapping:-

stall until ((Syncpt [<Indx>][31:0] >= (Payload[31:0] + Syncpt_Base[<Base_Indx>][31:0]))

13.3.2.13 LOAD_SYNCPT_BASE_32 <Base_Indx(8)>

This method copies the value from the current channel's Channel_Syncpt_Payload register into the Syncpt_Base register specified through the Base_Indx field:-

Syncpt_Base[<Base_Indx>][31:0] = Channel_Syncpt_Payload[31:0]

13.3.2.14 INCR_SYNCPT_BASE_32 <Base_Indx(8)>

This method adds the value of the current channel's Channel_Syncpt_Payload register into the Syncpt_Base register specified through Base_Indx:-

Syncpt_Base[<Base_Indx>][31:0] += Channel_Syncpt_Payload[31:0]

13.3.2.15 STALLCOUNT

Stallcount_Hi, Stallcount_Lo, and Stallctrl are methods to control “stall counters” through the command FIFO. Details of stall counters are given later in this document.

13.3.2.16 XFERCOUNT

Channel_Xfer_Hi, Channel_Xfer_Lo, and Xferctrl are methods to control “xfer counters” through the command FIFO.

13.3.3 Context Switching

Context management of Host1x 1x-based modules will be completely under software control (there will be no automatic context switching done by hardware). This mean a module will never produce a context switch interrupt, it will always operate in auto-acknowledge mode

A module with multiple hardware contexts (currently 2D), will have separate context IDs for each context. Software will have to do a SetClass to give the context ID associated with the hardware context that it wants to use. The 2D hardware will not automatically allocate contexts; software needs to program exactly the context it wants to use. For modules with a single hardware context (e.g., 3D), software must explicitly manage the save and restore of the context state.

13.3.4 Behavior of SetClass

The SetClass does not mean implicit acquisition of a module’s ownership. It is possible to send commands simultaneously from more than one channel to the same module. If exclusive ownership of a module is required for software correctness, then **acquire_mlock** and **release_mlock** opcodes should be used, which acquire and release the **MLOCKn** semaphore bits. An **acquire_mlock** that fails will cause that channel's commands to stall until it wins subsequent MLOCK arbitration (arbitrations happen with each **release_mlock**).

Here is the summary of SetClass behavior:-

- SetClass simply instructs the channel hardware which module and context to use for the following commands.
- If multiple channels write simultaneously to the same class ID, and no MLOCKs are used, then the actual commands are interleaved in round-robin fashion (one command per channel).
- If multiple channels write simultaneously to different class IDs in the same hardware module (and no MLOCKs are used), then the actual commands are interleaved, but in blocks of N-cycle bursts (each burst of commands must be preceded with a CTXSW to the module giving the new context ID).

  N in this case is programmable number configured through **Ctxsw_Timeout_Cfg** register.

13.3.5 Sync Points (SYNCPTs)

A Sync point is a mechanism to synchronize between software and Host1x clients and also in between Host1x clients. This is implemented as 32-bit counters which are incremented by 1 whenever some predestinated condition (or event) occurs. When a counter reaches its maximum value, it wraps back to zero on the next increment.

Tegra 4 series processors have 32 sync points. Sync points are not permanently associated with a channel; sync point allocation is done by software RM during initialization time.

There are two basic ways for sync point increments: -

- When a CPU writes an index to the Host1x's "syncpt_cpu_incr" register.
- When a Host1x client has received an "incr_syncpt" method and the condition specified by this method has become true.

Synchronization using sync points can be done in the following ways:

- A CPU can be interrupted when a sync point reaches a pre-specified value.
- A Host1x channel can have "wait" commands so that channel will wait for a pre-specified sync point value.

Sync points are normally not reset, but can wrap -- the comparison takes into account the possibility of wrapping.

Note:	Sync point wrapping works only if (Syncpt value - Syncpt Threshold) <= $2^{(syncpt\_width-1)}$ For 16-bit syncpt comparisons, the difference should be less than or equal to 32768. Software must take care of wrapping issues. For 32-bit comparisons, the difference should be less than or equal to 2,147,483,648, Because of the large value, software will not see any wrapping issues.

All Host1x clients (e.g., VI) implement the following increment syncpt method:

Incr_Syncpt <Condition> <Index>

The Host1x client would receive the "Incr_Syncpt" method and store the index for each condition. Whenever the "condition" event occurs, the client would return the index back to Host1x.

13.3.5.1 Client Model for Syncpt Behavior

Software Programming Model

The basic programming model that software will follow is:-

Each module will be programmed to do a unit of work (an operation) by Host1x using CDMA and push buffers.

Examples of an operation include:-

- BLT (VIC)
- Draw a set of triangles (GPU)
- Encode a single frame (MSENC).

If nothing else is programmed, then module will go idle until Host1x sends commands to start another operation (no continuous mode).To do its operation, a module reads data from memory and writes the results to memory. Modules interact with each other using memory buffers: one module is the producer of data, and another is the consumer of that data.

There are exceptions to this model which we will discuss in detail later, but should be mentioned here. The exceptions include VI and DISPLAY which work in continuous mode (they continue to process data without the need of additional Host1x programming).

Basic Synchronization

There are two basic needs for synchronization: management of memory buffers and timing of control register writes.

Memory buffers used to pass data from one module to the next use a producer/consumer model with circular buffers. To prevent buffer underflow and overflow, synchronization needs to be done in both directions:

- The consumer cannot read until producer is done writing (and the writes are committed to memory).
- The producer cannot reuse an output buffer (i.e., write to buffer) until the consumer is done reading the buffer.

Thus, the synchronization events required for memory buffers are:

- The module has completed all reads from the buffer.
- The module has completed all writes to the buffer (and they are committed by the memory controller).

To understand the requirements for timing the writes to control registers, a typical sequence is provided below:

1. reg wr for operation A
2. reg wr for operation A
3. reg wr (trigger) for operation A
4. reg wr for operation B
5. reg wr for operation B
6. reg wr (trigger) for operation B

If no WAIT method is placed between the trigger for A and the first register write for B, then in the worst case, corruption of operation A may occur because the new value of the control register is used before operation A is done. For modules that protect against this corruption, there is still the undesirable behavior of the module delaying the register write and subsequently causing back pressure on the Host1x write bus. If we wish to allow direct register reads to be used by ISRs they will happen asynchronously to the channel command writes, so one must ensure that the Host1x bus does not stall for significant periods of time.

For synchronizing writes to control register, a safe time to start writing register for the next operation is defined to be when:

- No corruption will occur for previous operations.
- No stall of the HWRBUS bus will result.

13.3.5.2 Standard Set of Incr_Syncpt Conditions

The following values of the “incr_syncpt cond” field are predefined:

- **0 (**Immediate): Return indx to Host1x immediately (used by software push-buffer allocation and helpful for debug).
- **1 (**Op_Done): Return indx to Host1x when all previously triggered operations have completed and their writes to memory are committed.
- **2** (Rd_Done): Return indx to Host1x when all previously triggered operations have completed their reads from memory.
- **3 (**Reg_Wr_Safe): Return indx to Host1x when it is safe to program registers for the next operation. Safe means no corruption to previous operations and no stalling of Host1x write bus will occur.

There are special cases which would require use of additional condition values. Some modules have multiple read buffers condition = 2 (all reads done) would mean all reads to all buffers done, but it might be useful to software to know when reads are done to a specific input buffer. For some modules, e.g., VI, there are different safe times to update different sets of registers, so condition = 3 will need several variations (one for each "safe time"). If a module can have two operations happening at one time (as in VI), then special considerations will need to be made for these cases: either two separate Incr_Syncpt methods, or one Incr_Syncpt method with many special conditions.

For some modules, one condition can replace another if the two conditions always happen within a short period of time of each other. Then the condition that always happens last can be used for both.

13.3.5.3 Continuous Mode — Display and VI / EPP

For each display, software wants a free-running syncpt increment on every display “Vsync”. The most appropriate implementation of this is to have an additional register which has “Enable” and “Indx” fields to control the increment of syncpt on every “Vsync” event. If enable is set, then on every “Vsync”, display would return this “Indx” back to Host1x.

A similar situation exists for VI, where besides a set of conditions for the “Incr_Syncpt” method, it would also need a register with “Enable” and “Indx” which would control the incrementing of a syncpt on every camera “Vsync”.

Host1x clients will implement the logic associated with these registers. When this continuous mode is enabled, client should set a pending bit for when the condition occurs. When the pending bit is set, client will arbitrate for the hrd_<client>2host1x bus and if selected it will send “Indx” tagged with “Syncpt type” on hrd_ bus. After successfully sending the indx on hrd bus, client will clear the pending bit.

One of the issues for continuous operation of VI and EPP is that not all modes of operation allow the consumer of their output buffers to signal backpressure when the consumer has fallen behind in reading. To alleviate this problem, VI and EPP have added registers which, when written to, signal that the reading of one buffer has completed.

13.3.5.4 Allowing Multiple Pending INCR_SYNCPTS

Clients can have multiple pending syncpts which are queued into the client’s syncpt FIFOs, which are dedicated ones for each syncpt conditions.

The behavior of these syncpt FIFOs is controlled by an “Incr_Syncpt_Cntrl” register in client’s register space:-

INCR_SYNCPT_CNTRL< No_Stall>< Soft_Reset>

- No_Stall:

  If this bit is 1 and an Incr_Syncpt method is received but the syncpt FIFO for that condition is full, then this method will be dropped and the Incr_Syncpt_Error[Cond] bit will be set.

  If this bit is 0, then instead of dropping next method in case of a syncpt FIFO full condition, the client will just stall the host interface.
- Soft_Reset:

  if Soft_Reset is set, then all internal state of the client syncpt block will be reset. To do a soft reset, first set Soft_Reset of all Host1x clients affected, then clear all Soft_Resets.

INCR_SYNCPT_ERROR<Cond_Status>

This register stores error status in case of above mentioned syncpt FIFO full condition.

13.3.5.5 32-bit Sync Point Comparison

In previous chips, sync point counters were implemented as 32-bit registers but comparison takes into account 16-bit value only. In Tegra 4 devices, sync point comparison is extended to 32 bits. To support this feature, Host1x has the following changes:-

Host1x Register Changes

Following are the registers that are extended to 32 bits.

- Syncpoint base register [Host1x 1x_Sync_Syncpt_Base]
- Syncpt Threshold register [Host1x 1x_Sync_Syncpt_Int_Thresh]

Following is a new 32-bit register to store payload value required in syncpt methods:

- Channel_Syncpt_Payload[31:0]

Host1x Method Changes

To support 32-bit syncpt comparisons, new Host1x methods have been introduced and are described later in this document.

13.4 Unit Description

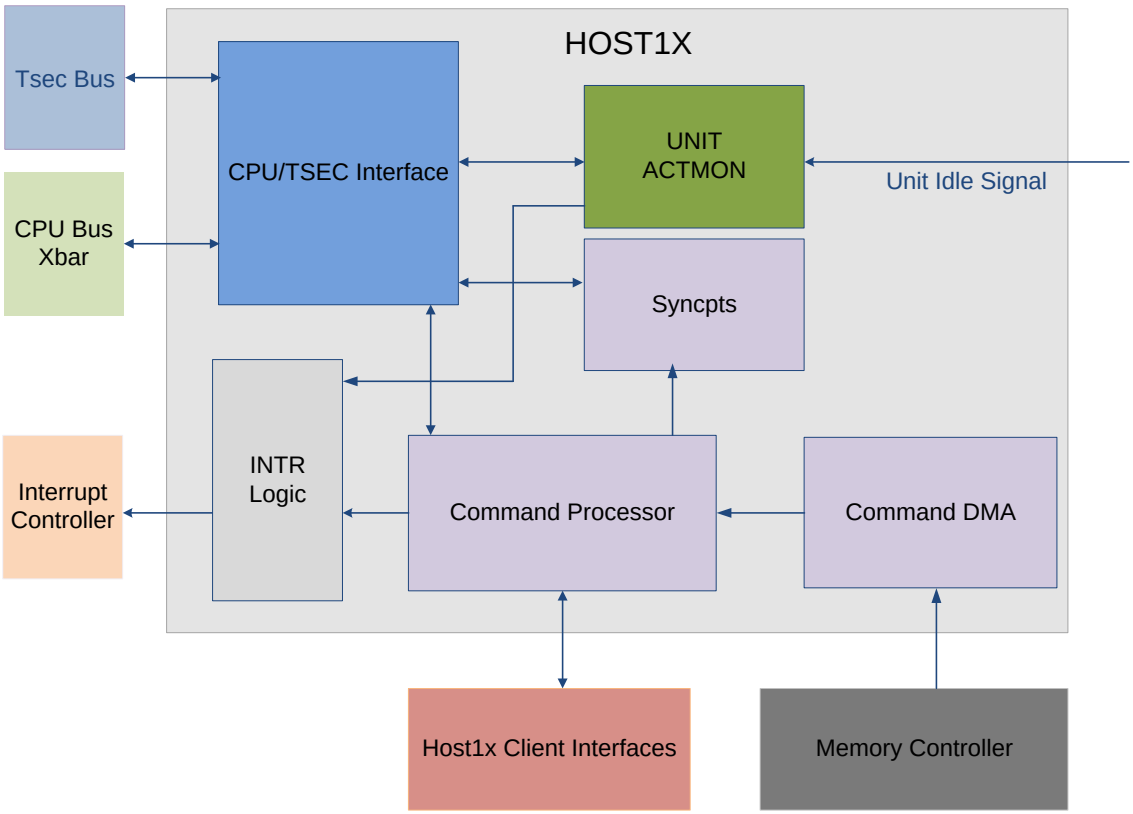
Host1x has master client interface with the CPU xbar bus and also TSEC unit to receive CPU/TSEC read/write commands. The incoming commands are either processed inside Host1x if they are Host1x specific or routed to clients.

Host1x has a point-to-point interface with its clients using HWR/HRD buses, which are used for sending requests and accepting responses from clients.

There is a memory controller interface to fetch Push-Buffer commands from memory through the “Command DMA” unit. The incoming Push-Buffer commands are processed inside “Command Processor” and afterwards either consumed inside host1x or it will generate tractions to client interface. There is an internal “syncpt” unit which is used to synchronize between Host1x clients and also between software.

There is a “Unit ActMon” block to monitor activities inside the GR3D unit which is used by software for power management.

Figure 35: Host1x Top-Level Block Diagram

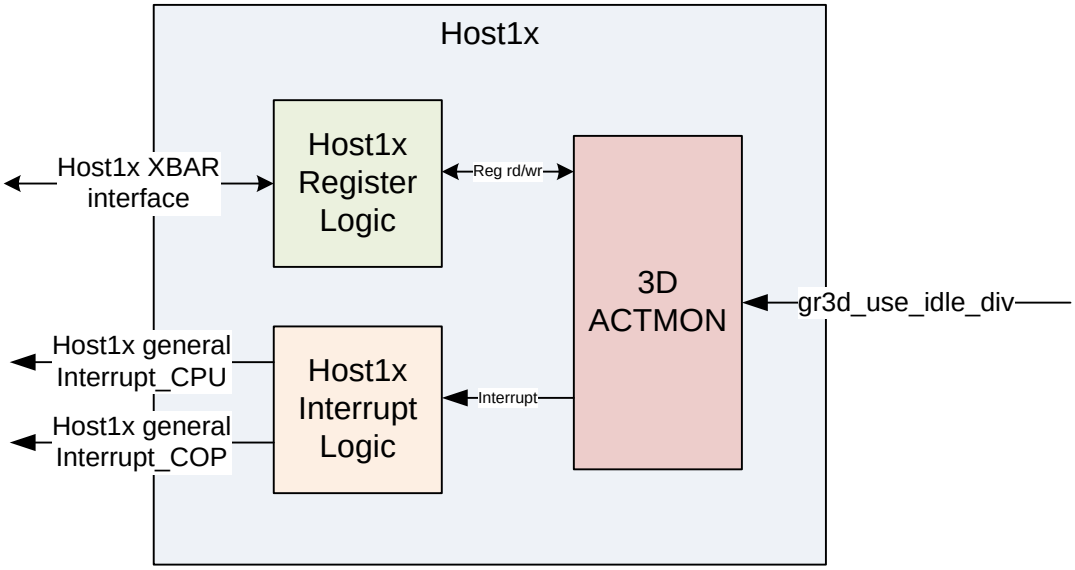


13.4.1 Activity Monitor for the 3D Unit

The Host1x unit implements 32-bit activity monitor counters to monitor the idleness of the 3D unit. The reason for choosing the Host1x unit is due to fact that register read latency is less in case of Host1x when compared to 3D unit.

The 3D engine sends idle/active information to Host1x through same “idle” signal which goes from 3D unit to CAR (which is a level signal). These activity monitor counters are a standard plug-in from activity monitor block. The activity monitor block provides averaging, watermark detection, histogramming, and interrupts functionality similar to the central ActMons. Host1x driver is changed to manage these monitors.

Figure 36: 3D ACTMON Interface



The 3D ACTMON register is a part of Host1x register space and can be accessed through native Host1x direct/indirect read and write. The interrupts from the 3D ACTMON unit are ORed with other Host1x general interrupt sources and passed on to the Host1x 1x_general_interrupt_cpu/cop output pin. The Host1x interrupt status register (HINTSTATUS) is modified to include an additional bit to indicate if the interrupt source is 3D ACTMON.

Table 45. Actmon Interrupt Status Register

Register([Bit]) Name	Description
HINTSTATUS[9]	Set by 3D ActMon (software needs to read HOST1X_gr3D_actmon Interrupt Status Register to determine the exact source of interrupt) When this bit is set, the HINTSTATUS_EXT_INT bit will also be set in the HINSTATUS register. This bit is sticky and PENDING until cleared (cleared by writing 1)

There is a corresponding mask bit inside Host1x interrupt registers.

13.4.2 Timeout Mechanism

Host1x has a mechanism to generate interrupt for unserviced CPU read/write requests which is controlled through a timeout register (HOST1X 1x_IP_TIMEOUT). Host1x will generate a timeout Interrupt and violating address is stored in following registers:-

- IP_READ_TIMEOUT_ADDR
- IP_WRITE_TIMEOUT_ADDR

In earlier chips, this timeout register is 16 bits, which is not sufficient because 3D unit read latency can be greater than 320 µs. So the width of the timeout register has been increased to 32 bits to provide a timeout value of ~19 seconds.

The Tsec master client interface also uses this mechanism to generate timout interrupt to CPU in case of illegal read/write access. HINTSTATUS_EXT register has been modified to indicate if the timeout is from native CPU access or TSEC access.

Also in case of TSEC read timeout the returned response packet will have an additional bit (tsec2host1x 1x_iprdata_timeout) to indicate a timeout error.

HINTSTATUS[Timeout_source=bit10] =0 Timeout from TSEC access.

1 Timeout from XBAR access.

The current scheme saves only the last timeout address in case of multiple timeout happens before clearing the timeout interrupt by software.

13.4.3 Performance Statistics Counters

Host1x has performance counters for profiling, which can be enabled independent of push-buffer commands. Host1x needs the following statistics per channel:

- Total clocks
- Clocks stalled waiting for syncpt
- Clocks transferring data

13.4.3.1 Tick Counter

Each channel has its own 64-bit tick counter that is incremented on each Host1x clock, given that it is enabled. TICKCTRL enables and disables the tick counter.

The TICKCOUNT_LO and TICKCOUNT_HI 32-bit registers can be written to initialize the tick counter. Reads of these registers return the tick count. TICKCOUNT_LO stores the lower 32 bits, and TICKCOUNT_HI stores the upper 32 bits. TICKCOUNT_HI is calculated as follows:

```
if (TICKCOUNT_LO == 0xFFFF_FFFF && TICKCTRL == enable) TICKCOUNT_HI++;
```

13.4.3.2 Channel Stall Counter

This is a 64-bit CHANNEL_STALL counter per channel, which when enabled will get increment on each clock cycles if that particular channel is waiting for syncpt.

Following are the registers to control this counter:

- STALLCTRL : - This is a 1-bit register to enable/disable CHANNEL_STALL counter.
- STALLCOUNT_LO :- This is a 32-bit register to initialize lower 32 bit of CHANNEL_STALL counter.
- STALLCOUNT_HI :- This is a 32-bit register to initialize upper 32 bit of CHANNEL_STALL counter.

Following are the new Host1x class methods to program this counter:

- STALLCOUNT_HI:- This method will initialize the high 32 bits of
  - tick count value in CHANNEL_STALL counter.
- STALLCOUNT_LO :- This method will initialize the low 32 bits of
  - tick count value in CHANNEL_STALL counter.
- STALLCTRL:- This method will enable/disable CHANNEL_STALL counter.

13.4.3.3 Channel Xfer Counter

This 64-bit CHANNEL_XFER counter per channel measures the data transfer interval per channel. This will be incremented at each clock cycles if channel is not idle (busy in fetching channel commands).

Following registers will be used to initialize this counter:

- XFERCTRL: This 1-bit register enables/disables the CHANNEL_XFER counter.
- CHANNEL_XFER_LO[31:0]: This 32-bit register initializes the lower 32 bits of the CHANNEL_XFER counter.
- CHANNEL_XFER_HI[31:0]: This 32-bit register initializes the lower 32 bits of the CHANNEL_XFER counter.

Following are the new Host1x class methods to program this counter:-

- CHANNEL_XFER_HI: This method will initialize the high 32 bits of the tick count value in the CHANNEL_XFER counter.
- CHANNEL_XFER_LO: This method will initialize the low 32 bits of the tick count value in the CHANNEL_XFER counter.
- XFERCTRL: This method will enable/disable the CHANNEL_XFER counter.

Note: Counter operations are identical irrespective of whether they are controlled through registers or other methods.

13.4.4 Interrupts

Host1x has following types of interrupts present:

13.4.4.1 Host1x Client Interrupt

Host1x is the collection point for all of its clients interrupts. Also it has control over those using status and mask registers with fields for each client. These client interrupts are forwarded to the central interrupt controller unit, individually through per client interrupt lines.

13.4.4.2 Host1x Interrupts

Host1x -specific interrupts are specified through the HOST1X_SYNC_HINTSTATUS_0 register while additional interrupts have been specified through the HOST1X_SYNC_HINTSTATUS_EXT_0 register (see Registers below).

13.4.4.3 Host1x Syncpt Interrupts

Host1x can generate a syncpt interrupt upon reaching a threshold value by syncpt registers. It can also generate an interrupt to either CPU0, which is the main CPU complex, or CPU1, which is the AVP.

An interrupt is routed to cpuN when:

(SYNCPT[indx] >= SYNCPT_INT_THRESH[indx]) && (SYNCPT_THRESH_INT_MASK[indx] == cpuN)

The comparison takes wrapping into account. See the “Sync Points (SYNCPTs)” section for more information on wrapping.

The status is sticky and is PENDING until cleared. Write 1's to clear.

13.4.5 Indirect Register Access

Host1x provides a means of indirect register access to all its clients. An indirect register access involves multiple steps and requires a channel for the read return data. All registers required for indirect accesses resides in a channel.

Writes involve two Host1x register writes in the owning channel's space; the first write indicates the address (INDOFF, or INDOFF2 and INDCTRL), and the second indicates the data (INDDATA).

Reads involve two direct Host1x writes, the first indicates the address (again INDOFF, or INDOFF2 and INDCTRL), and the second to indicate the number of reads (INDCNT). It also requires an indeterminable number of direct Host1x reads to the read return FIFO status register followed by a read to the read return FIFO.

- INDOFF
- INDOFF2
- INDCTRL
- INDDATA
- INDCNT

Pitfalls — Indirect accesses require a software mutex to ensure that no other software threads access the indirect registers while issuing the access (since the series of accesses is non-atomic). Indirect reads can only issue counts less than the read return FIFO size to prevent a deadlock – too many read requests can block popping of the read return FIFO. This restriction may be even tighter. All modules accesses must also be tied to a channel.

13.4.6 Direct Register Access

All clients under Host1x have 256KB of address space, which originates from the space specified by indirect offset register (INDOFF). Host1x's client address map is dictated by its client's module IDs; these IDs are used in INDOFF to specify the target module of the register access. A client's address in the Host1x space is calculated as:

address = direct_access_base_address(0x54000000) + (module_id << 18)

+ 4 * register_offset

There can only exist one pending read per interface. Reads are returned from the client when the client is ready. There are no latency restrictions.

Indirect versus Direct — pitfall of direct addressing is variable latency of Host1x's clients. While this has no impact on writes, the CPU is blocked until the read returns rendering the CPU idle during that time.

13.4.7 Host Clients

The following table lists the clients that are accessed via Host1x with their module (client) ID, which is used to determine addresses for direct addressing.

Table 46: Host1x Clients

Client	ID
Host1x	0x0
VI	0x2
CSI	0xf
EPP	0x3
ISP	0x4
2D	0x5
3D	0x6
Display A	0x8
Display B	0x9
HDMI	0xa
DSI	0xc
DSIB	0x10
MSENC	0x13
TSEC	0x14

13.4.8 Class IDs

The following table lists the class IDs.

Class	ID
NV_HOST1X_CLASS_ID	0x01
NV_VIDEO_ENCODE_MSENC_CLASS_ID	0x21
NV_VIDEO_STREAMING_VI_CLASS_ID	0x30

Class	ID
NV_VIDEO_STREAMING_EPP_CLASS_ID	0x31
NV_VIDEO_STREAMING_ISP_CLASS_ID	0x32
NV_GRAPHICS_2D_DOWNLOAD_CLASS_ID	0x50
NV_GRAPHICS_2D_CLASS_ID	0x51
NV_GRAPHICS_2D_SB_CLASS_ID	0x52
NV_GRAPHICS_2D_DOWNLOAD_CTX1_CLASS_ID	0x54
NV_GRAPHICS_2D_CTX1_CLASS_ID	0x55
NV_GRAPHICS_2D_SB_CTX1_CLASS_ID	0x56
NV_GRAPHICS_2D_DOWNLOAD_CTX2_CLASS_ID	0x58
NV_GRAPHICS_2D_SB_CTX2_CLASS_ID	0x5A
NV_GRAPHICS_3D_CLASS_ID	0x60
NV_DISPLAY_CLASS_ID	0x70
NV_DISPLAYB_CLASS_ID	0x71
NV_HDMI_CLASS_ID	0x77
NV_DISPLAY_DSI_CLASS_ID	0x79
NV_DISPLAY_DSIB_CLASS_ID	0x7A
NV_TSEC_CLASS_ID	0xE0

13.4.9 Host Address Space

Each channel is allocated 16KB of space, and they are aligned contiguously at the top of the Host's address space. A channel's space consists of channel registers, the command FIFO, and an aliased frame buffer region, which are unique per channel. The rest of the space consists of synchronous registers, which are aliased in each channel – there exists only one copy.

13.4.9.1 Channel Map

The following table gives offsets from the channel base address for different sets of registers available to each channel.

Table 47: Host1x Channel Map

0x0000	Channel Registers
0x0600	Reserved (former RDMA Registers)
0x0800	Command FIFO
0x1000	Frame Buffer
0x2000	Reserved (former Asynchronous Registers)
0x3000*	Synchronous Registers
0x3FFF	Bottom

* 0x3000 is the synchronous registers base (HOST1X_CHANNEL_SYNC_REG_BASE)

13.4.9.2 Host1x Map

The following table shows the Host1x addresses for CPU/COP. Channels starts at the lowest addresses and are aligned contiguously. Following the channels there is a protected channel, whose map looks like that of a channel's (although some of the registers are not functional, more on the protected channel later). Direct access to Host clients starts at 5400_0000, but this is configurable through a BAR register.

Table 48: Host1x Map

5000_0000-5000_3FFF	Channel 0 (16KB)
5000_4000-5000_7FFF	Channel 1 (16KB)
5000_8000-5000_BFFF	Channel 2 (16KB)
5000_E000-5001_2FFF	Channel 3 (16KB)
5001_3000-5001_6FFF	Channel 4 (16KB)
5001_7000-5001_AFFF	Channel 5 (16KB)
5001_B000-5001_BFFF	Channel 6 (16KB)
5001_C000-5001_FFFF	Channel 7 (16KB)
5002_0000-5002_3FFF	Channel 8 (16KB)
5002_4000-5002_7FFF	Protected Channel base

13.5 Performance

13.5.1 Key Use Cases

Command throughput must be sufficient when all channels are active, the CDMA is active, and spooling is active.

13.5.2 Latency Targets

Direct read latency must be under 1 ms. Clients that cannot guarantee that a read will return in under 1 ms must not be read directly but indirectly.

13.5.3 Bandwidth Targets

As Host1x unit is used only for sending control commands to clients and usually at frame interval.

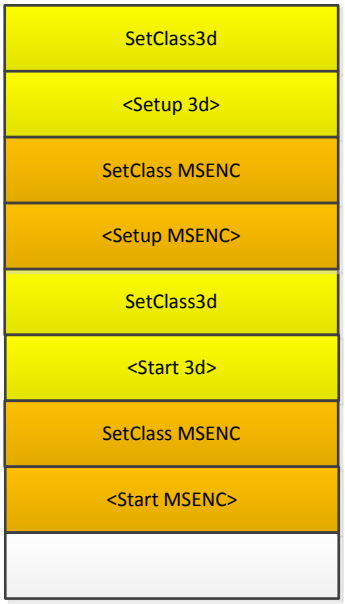
As these control commands are short (~200 -300 words maximum), it is expected that the bandwidth requirement on each client should be less than 1MB/s (3MB/s in case of 3D engine), whereas the maximum available bandwidth per client on the HWR bus will be (206*4B)/12 =68MB/s at 206 MHz Host1x clock. The minimum bandwidth on the HWR bus per client is 22 MB/s at 66 MHz clock which should easily meet the client requirement.

13.6 Host1x Programming Model

The Host programming models resides on top of the concept of channels. A channel provides the means for software to supply an ordered sequence of commands to one or more classes. There are no restrictions on how many classes a channel may own or how often it can switch between classes, but it can only operate on one class at a time (known as the working class) and one command at a time. A stalled channel does not allow any commands to proceed from any class – there is a strict ordering of commands. A channel starts processing commands simply when commands are present, either supplied by the CPU or gathered from memory. A channel can be stopped explicitly by a CPU write to channel state.

A single channel owning multiple classes and clients:

Figure 37: Single Channel Example



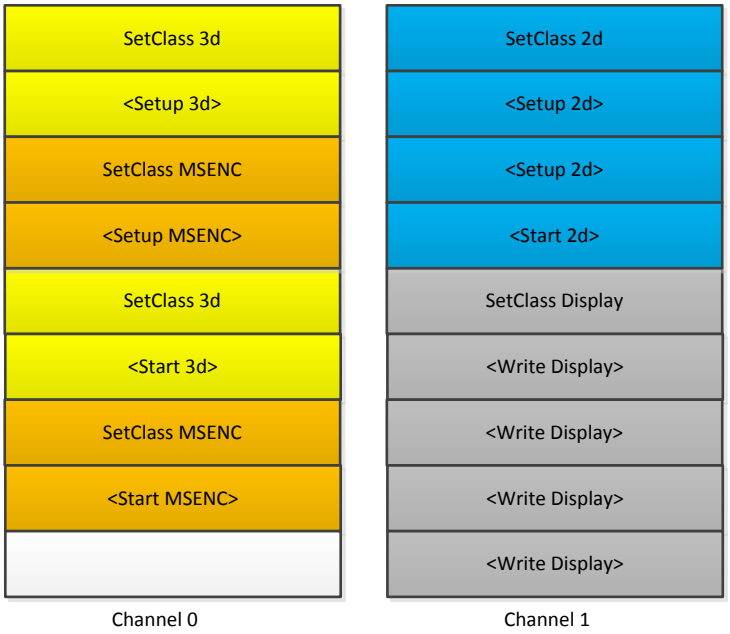
13.6.1 Concurrency

Channels operate in parallel and are unhindered by one another except in two specific cases:-

- Synchronization points
- Class contention

The following figure depicts such concurrency:

Figure 38: Channel Concurrency Example



In the example above, no channel is blocked by another at any time so both can work concurrently. Care must be taken when assigning clients to channels. Channel concurrency as well as channel throughput is crucial for performance. For example, a client that is event-driven with a low latency requirement should exist in its own channel. Conversely, two clients that must be steadily supplied with commands could possibly coexist in the same channel given that their synchronization points are not conflicting (if they have any real synchronization points at all). Clients that operate sequentially on the same piece of data can easily reside in the same channel.

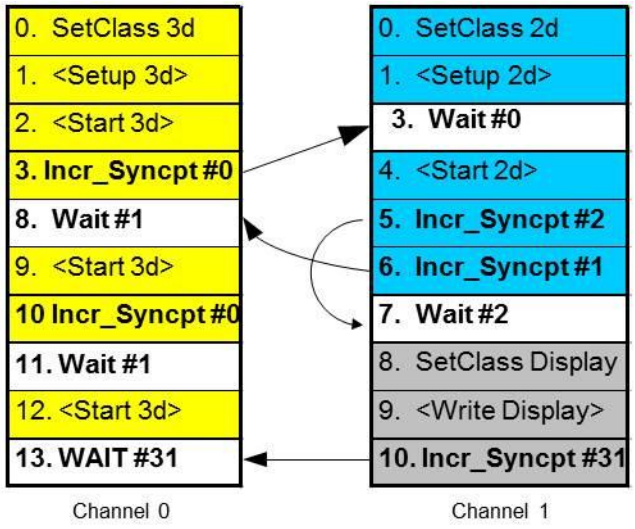
Clients with low-latency requirements like described in the first example above should possibly be moved to a PIO programming model and away from a channel-based model.

13.6.2 Synchronization

There are 32 unique synchronization points (syncpts) available per channel. Similar to semaphores, syncpt methods are issued to clients, mentioning syncpt counter index and returning condition. Any channel can be made to wait on a syncpt value of a particular syncpt index through Host1x “wait” method. Once the syncpt method is received by clients, based on syncpt condition, they will return a syncpt index back to Host1x. On receiving a syncpt index from the client, Host1x will increment that particular syncpt counter.

The following simple example below details how syncpts and waits allow for synchronization. The arrows simply indicate channel dependencies, which channel is waiting for syncpt increment from other channel.

Figure 39: Channel Synchronization Example



13.6.3 Progress Status

Channel progress is monitored through two means – GET and syncpts. GET is a channel state that indicates the address of the last command that has executed in the channel. GET is the same as DMAGET.

Syncpts are counter registers that are incremented when specified events occur (e.g., after the completion of each operation done by a module). The 32-bit syncpt values typically are monotonically increasing and can be used for in channel waits and out-of-channel interrupts.

The GET and syncpt registers can also be read directly (out-of-band) by the CPU.

13.6.4 Syncpt Base Register Use Case

The syncpt base registers are modified through LOAD_SYNCPT_BASE_32 or INCR_SYNCPT_BASE_32 methods and are used with the WAIT_SYNCPT_BASE_32 method.

This mechanism is useful when software wants to wait for a particular syncpt increment, but software does not know the absolute value of the sync point register until later.

Definition: the "future value" of a sync point is the value that the syncpt register will contain right after a particular increment has occurred.

Consider this example:-

In this example, time increases downward (later (lower) lines are ahead in time).

Software "knows" the syncpointA future-value equals N at this point; software "knows" the baseA future-value equals N at this point

1. command1
2. increment syncpointA to N+1
3. command2
4. increment syncpointA to N+2
5. command3
6. wait until command1 is done (syncpoint == N+1)

7. command4
8. wait until command2 is done (syncpoint == N+2)
9. command5

If software actually DOES know the value of N then regular WAIT_SYNCPT works fine here. The problem is that the user space software driver (which is creating this list of commands) does not know what the value N is until after these commands (1-9) are all flushed to the kernel space driver.

The kernel space driver receives packets of commands from many different user space processes. When it receives a packet of commands it queues those commands (i.e., writes the Host1x PUT pointer). It also keeps track of all increments that have occurred in all commands that have been queued so far. The "future-value" of each sync point is simply the total number of increments that have occurred since the beginning of time (actually "beginning of time" is really the point in time where the kernel space driver was initialized and reset all the syncpt registers to 0).

So when the kernel space driver queues these commands (1-9), it knows the value N (i.e., the sum of all increments since the beginning of time). It can pass this back to the user space driver. But the user space driver does not know how many increments will be queued before commands 1-9 are queued because any other process can queue commands (including increments) at any time (e.g., after the user space driver has written commands 2 and 4 to the buffer, but before those commands have been queued by the kernel driver).

To solve this, we add a command at the end of each packet that increments the base register the same number of times as the sync point was incremented:

10. baseA += 2

This means that, at the start of any packet of commands, the future value of syncpointA and the future value of baseA are always the same. So command 6 can be:

WAIT_SYNCPT_BASE syncpt=A base=A offset=1

and command 8 can be:

WAIT_SYNCPT_BASE syncpt=A base=A offset=2

Note: Software never writes a new value to the syncpointA or baseA register except when the kernel driver is initialized (e.g., when the system boots). After that we only increment the syncpoint register and only add to the base register. All of this depends on all software that creates command buffers cooperating. Software has this policy (e.g., the user space driver always tells the kernel driver how many increments are contained in each packet of commands)

13.6.5 Indirect/Direct Register Addressing Scheme

The indirect addressing scheme works as follows:

- An address is decoded and either routed to a channel register or to a synchronous register
- A write to INDCNT or INDDATA triggers a register access to the register specified in INDOFF
- A transaction is created and pushed into the REGF FIFO. The owning channel is specified by which INDCNT or INDDATA is written. The client and offset are specified in INDOFF.
- The transaction is routed to the target client over the HWR bus.
- If the access is a write, the transaction is complete. In the case of a read, the processor polls the channel's read return FIFO status (FIFOSTAT), waiting for a nonempty FIFO.
- The client returns the read to the channel's return FIFO.
- The processor reads INDDATA register of that particular channel to pops data from the return FIFO.

Direct addressing uses the same flow with some adjustments:

- An address is decoded and either routed to a channel register, a synchronous register, or a client
- This step is nonexistent; direct accesses do not require a trigger.
- In the case of a client decode, a transaction is created and pushed into a FIFO called REGF. The owning channel is specified as CPU_READ_RETURN_TAG. The client and offset are indicated by the address: the client is created by shifting the address down by 18; the offset is simply the lower 18 bits.
- The transaction is routed to the target client over the HWR bus. If the access is a write, the transaction is complete.
- In case of a read, client returns the read data and channel ID in the returned packet indicates its destination, be it interface or a channel's return FIFO. If the channel is 0-8, it is an indirect read and returned to the indicated channel. If the channel matches CPU_READ_RETURN_TAG(0xf), the data is returned to the IP interface.

Currently, there is support for only one pending read per interface.

13.7 Host Channel Opcodes

This section providesthe format of opcodes that can be sent through the command FIFO.

HCFCMD

Generic command FIFO packet (contains fields common to all opcodes) and is used for initial decode. All command FIFO packets are multiples of 32 bits.

HCFSETCL

The SetClass opcode is to specify which class is being to be referenced (may cause rerouting of subsequent methods/data). In addition to switching classes, the opcode allows some methods to be programmed on the switch similar to a HCFMASK opcode.

HCFINCR

The Incrementing opcode indicates the offset should be incremented, for each data that is part of the packet. The count argument indicates how many 32-bit values are following. If channel protect is enabled, it means the host should prevent a channel switch from occurring at the end of this command packet.

HCFNONINCR

The Non-Incrementing opcode indicates the same offset should be sent for each data that is part of the packet. The count argument indicates how many 32-bit values are following. If channel protect is enabled, it means the host should prevent a channel switch from occurring at the end of this command packet.

HCFMASK

The Mask opcode, from the starting offset, generates offsets based on where the bits are set in the mask. The host expects the amount of data following to equal the number of bits set. If channel protect is enabled, it means the host should prevent a channel switch from occurring at the end of this command packet.

HCFIMM

The Immediate opcode indicates the offset and data are contained in the same 32-bit datum. Only the lowest 16 bits of data are sent to the module (IMMDATA). The upper 16 bits are zeroed out.

HCFRESTART

The Restart opcode is specific to DMA operation and causes the host to set DMAGET to (ADDRESS << 4), so the next command fetch will be from (DMASTART + DMAGET).

In legacy chips, bits 27:0 were not decoded and assumed to be 0's (allowing only simply wrapping of GET back to the top of the command buffer). ADDRESS can be 0 for compatible RESTARTs or non-zero acting as a JUMP.

Note that the jump address granularity is 16 bytes, since the bottom 4 bits cannot be specified.

HCFGATHER

The Gather opcode allows contiguous portions of memory to be fetched and placed in line with the command stream, replacing the 2 words of the gather command. It optionally can put an incrementing or non-incrementing opcode in the stream ahead of the gathered data. This allows for the gathered data to be a pure data stream and not be required to have host opcodes inside.

HCFCHDONE

This opcode indicates to the command processor that the current channel is done processing for now and is willing to give up any of its owned modules to other channels that need them.

13.8 Host Channel Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

13.8.1 HOST1X_CHANNEL_FIFOSTAT_0

CFNUMEMPTY is the number of free slots available in the per-channel command. A FIFO is needed for PIO or polling for completion of a wait.

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

Bit	Reset	Description
31	X	INDRDY: Indicates that INDCOUNT==0, so it should be OK to issue another read
28:24	X	OUTFENTRIES: Number of entries available for reading in this channel's output FIFO
20:16	X	REGFNUMEMPTY: Register write/read FIFO free count
13	X	CFGATHER3D: Indicates whether GATHER3D is active. If a GATHER3D command issued via PIO, software must wait for the GATHER3D to be IDLE before issuing another command. 0 = IDLE 1 = BUSY
12	X	CFGATHER: Indicates whether GATHER is active. If a GATHER command issued via PIO, software must wait for the GATHER to be IDLE before issuing another command. 0 = IDLE 1 = BUSY
11	X	CFEMPTY: Indicates whether the command FIFO is empty or not 0 = NOTEMPTY 1 = EMPTY
10:0	X	CFNUMEMPTY: Command FIFO free count

13.8.2 HOST1X_CHANNEL_INDOFF_0

The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA.

The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set the other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases. For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.

Indirect frame buffer writes are STRONGLY DISCOURAGED. There are better ways to write to memory (direct and through the channel memory map) and there is limited flow control in the host. It is very easy to get into trouble with indirect frame buffer writes.

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTOINC: Auto increment of read/write address 0 = DISABLE 1 = ENABLE
30	X	ACCTYPE: Access type: indirect register or indirect frame buffer 0 = REG 1 = FB
29	X	BUF32B: Buffer up 32 bits of register data before sending it. Otherwise, register writes will be sent as soon as they are received. Does not support byte writes in 16-bit host. Does not affect frame buffer writes. 0 = NOBUF 1 = BUF
28:27	X	INDSWAP: Indirect frame buffer access swap control. 00 = No byte swap 01 = 16-bit byte swap ([31:0] -> {[23:16],[31:24],[7:0],[15:8]}) 10 = 32-bit byte swap ([31:0] -> {[7:0],[15:8],[23:16],[31:24]}) 11 = 32-bit word swap ([31:0] -> {[15:8],[7:0],[31:24],[23:16]}) 0 = NONE 1 = BYTE16 2 = BYTE32 3 = WORD32
25:18	X	INDMODID: ACCTYPE=REG: Register module ID 0 = HOST1X 1 = MPE 2 = VI 3 = EPP 4 = ISP 5 = GR2D 6 = GR3D 8 = DISPLAY 9 = DISPLAYB 12 = DSI 10 = HDMI 16 = DSIB 19 = MSENC 20 = TSEC
25:2	X	INDOFFSET: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET: ACCTYPE=REG: register offset ([15:0])
0	X	INDOFFUPD: Optionally disable the update of INDOFFSET when writing this register 0 = UPDATE 1 = NO_UPDATE

13.8.3 HOST1X_CHANNEL_INDCNT_0

Indirect Register Access Count

Used to trigger indirect reads. Holds the number of registers/memory locations that will be read out. Channels should not request more than there is space available in their output FIFO. Only the protected channel should make liberal use of this feature for speeding up context switching.

For indirect frame buffer reads, each channel cannot issue more than NV_HOST1X_MAX_IND_FB_READS at once. The read data must return and be written into the per-channel output FIFO before any additional reads can be issued.

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	INDCOUNT

13.8.4 HOST1X_CHANNEL_INDDATA_0

This register, when written, writes to the data to the INDOFFSET in INDOFF. For reads, a REGFNUMEMPTY number of 32-bit values can be read before needing to poll FIFOSTAT again. The per-channel output FIFO (OUTFENTRIES) is readable via this offset. A read of INDDATA will pop an entry off of the per-channel output FIFO.

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INDDATA: read or write data

13.8.5 HOST1X_CHANNEL_RAISE_0

The general-purpose channels have DMA and RAISE/REFCOUNT functionality.

Any raise values returned from a client module are converted to vectors and update the per-channel raise register. The RAISE vector is also writable by the CPU. Any bits set in the RAISE field when written will be set in the RAISE register, allowing any pending WAITs to continue.

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	RAISE: This channel's RAISE vector

13.8.6 HOST1X_CHANNEL_DMASTART_0

This register triggers a DMA fetch from the frame buffer for this channel, if the Put register does not equal the DMA Get register.

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	DMASTART: cmdbuf frame buffer offset

13.8.7 HOST1X_CHANNEL_DMAPUT_0

This register triggers a DMA fetch from the frame buffer for this channel, if the PUT register does not equal the GET register. This address is relative to the DMASTART base address. Does not support byte writes. All 4-byte data need to be programmed.

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000xx)

Bit	Reset	Description
31:2	0x0	DMAPUT: cmdbuf frame buffer offset

13.8.8 HOST1X_CHANNEL_DMAGET_0

This register tracks the frame-buffer offset the DMA engine has read up to (incremented as entries are loaded from the channels command buffer into the FIFO). This address is relative to the DMASTART base address.

Offset: 0x1c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	DMAGET: cmdbuf frame buffer offset

13.8.9 HOST1X_CHANNEL_DMAEND_0

The boundary of illegal addresses (either end of push-buffer or end of physical memory). This is designed to prevent DMA from prefetching illegal addresses. If DMA reaches this address before seeing a RESTART, it will stop. This would be a software error condition.

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

Bit	Reset	Description
31:2	X	DMAEND: cmdbuf frame buffer offset

13.8.10 HOST1X_CHANNEL_DMACTRL_0

DMA Control Register

DMAGETRST: Reset GET pointer to '0'. Useful for cleaning up crashed channels. DMAGET is not updated instantly. Takes 4 cycles between programming of reset and a valid DMAGET.

DMAGETINIT: Reset the GET pointer to the value of DMAPUT when DMAGETRST is asserted.

DMASTOP: Stop DMA from fetching on this channel.

Note: A Command DMA channel needs to be enabled for PIO-gather to work.

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx001)

Bit	Reset	Description
2	0x0	DMAINITGET: Reset GET pointer to the value of DMAPUT when DMAGETRST is asserted. 0 = DISABLE 1 = ENABLE
1	0x0	DMAGETRST: Reset GET pointer to '0'. Useful for cleaning up crashed channels. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x1	DMASTOP: Stop DMA from fetching on this channel. NOTE: a Command DMA channel needs to be enabled for PIO gather to work. 0 = RUN 1 = STOP

13.8.11 HOST1X_CHANNEL_INDOFF2_0

The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA.

The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set the other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases. For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.

Indirect frame-buffer writes are STRONGLY DISCOURAGED. There are better ways to write to memory (direct and through the channel memory map) and there is limited flow control in the host. It is very easy to get into trouble with indirect frame-buffer writes.

Offset: 0x8c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:18	X	INDMODID2: ACCTYPE=REG: Register module ID 0 = HOST1X 1 = MPE 2 = VI 3 = EPP 4 = ISP 5 = GR2D 6 = GR3D 8 = DISPLAY 9 = DISPLAYB 10 = HDMI 12 = DSI 16 = DSIB 19 = MSENC 20 = TSEC
31:2	X	INDOFFSET2: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET2: ACCTYPE=REG: register offset ([15:0])

13.8.12 HOST1X_CHANNEL_TICKCOUNT_HI_0

This register holds the high 32 bits of the tick count value.

Offset: 0x90 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_HI

13.8.13 HOST1X_CHANNEL_TICKCOUNT_LO_0

This register holds the low 32 bits of tick count value.

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_LO

13.8.14 HOST1X_CHANNEL_CHANNELCTRL_0

This register will be used for controlling some channel-related commands including enabling/disabling of the tick counter.

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	ENABLETICKCNT: enable or disable tick counter 0 = DISABLE 1 = ENABLE

13.8.15 HOST1X_CHANNEL_PAYLOAD_0

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	PAYLOAD

13.8.16 HOST1X_CHANNEL_STALLCTRL_0

Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE_CHANNEL_STALL

13.8.17 HOST1X_CHANNEL_STALLCOUNT_HI_0

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	STALLCOUNT_HI

13.8.18 HOST1X_CHANNEL_STALLCOUNT_LO_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	STALLCOUNT_LO

13.8.19 HOST1X_CHANNEL_XFERCTRL_0

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE_CHANNEL_XFER

13.8.20 HOST1X_CHANNEL_CHANNEL_XFER_HI_0

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CHANNEL_XFER_HI

13.8.21 HOST1X_CHANNEL_CHANNEL_XFER_LO_0

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CHANNEL_XFER_LO

13.8.22 HOST1X_CHANNEL_HOST1X_CHANNEL_SPARE_0

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00ff0000 (0b00000000111111110000000000000000)

Bit	Reset	Description
31:16	0xff	CHANNEL_SPARE_HI
15:0	0x0	CHANNEL_SPARE_LO

13.9 Host SYNC Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

13.9.1 HOST1X_SYNC_INTSTATUS_0

INTSTATUS - interrupt status contains the interrupt status for all of the client modules. These status bits are only status. Writing '1' to them will not clear them. Software must clear the interrupt in the appropriate module, which should be reflected here.

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

Bit	Reset	Description
31	X	SYNCPT_CPU1_INT: set if SYNCPT_CPU1_INTSTATUS has a pending interrupt 0 = NOT_PENDING 1 = PENDING
30	X	SYNCPT_CPU0_INT: set if SYNCPT_CPU0_INTSTATUS has a pending interrupt 0 = NOT_PENDING 1 = PENDING
20	X	TSEC_INT: 0 = NOT_PENDING 1 = PENDING
19	X	MSENC_INT: 0 = NOT_PENDING 1 = PENDING
16	X	DSIB_INT: 0 = NOT_PENDING 1 = PENDING
12	X	DSI_INT: 0 = NOT_PENDING 1 = PENDING

Bit	Reset	Description
10	X	HDMI_INT: 0 = NOT_PENDING 1 = PENDING
9	X	DISPLAYB_INT: 0 = NOT_PENDING 1 = PENDING
8	X	DISPLAY_INT: 0 = NOT_PENDING 1 = PENDING
6	X	GR3D_INT: 0 = NOT_PENDING 1 = PENDING
5	X	GR2D_INT: 0 = NOT_PENDING 1 = PENDING
4	X	ISP_INT: 0 = NOT_PENDING 1 = PENDING
3	X	EPP_INT: 0 = NOT_PENDING 1 = PENDING
2	X	VI_INT: 0 = NOT_PENDING 1 = PENDING
0	X	HOST_INT: 0 = NOT_PENDING 1 = PENDING

13.9.2 HOST1X_SYNC_INTMASK_0

Contains a master interrupt mask for all interrupt signals. If the interface's MASK_ALL bit is disabled, no interrupts will be triggered on that interface. This applies to only the non-syncpt interrupts.

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	CPU1_INT_MASK_ALL: 0 = DISABLE 1 = ENABLE
0	0x0	CPU0_INT_MASK_ALL: 0 = DISABLE 1 = ENABLE

13.9.3 HOST1X_SYNC_INTC0MASK_0

INTC0MASK - Interrupt Mask for CPU0

Contains the interrupt mask bits for all of the client modules. If the INT_MASK is ENABLED for a particular module, the modules INT bit from INTSTATUS contributes to CPU0's interrupt signal.

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxx00xx0xxx0x000x00000x0)

Bit	Reset	Description
20	0x0	TSEC_INT_C0MASK: 0 = DISABLE 1 = ENABLE
19	0x0	MSENC_INT_C0MASK: 0 = DISABLE 1 = ENABLE
16	0x0	DSIB_INT_C0MASK: 0 = DISABLE 1 = ENABLE
12	0x0	DSI_INT_C0MASK: 0 = DISABLE 1 = ENABLE
10	0x0	HDMI_INT_C0MASK: 0 = DISABLE 1 = ENABLE
9	0x0	DISPLAYB_INT_C0MASK: 0 = DISABLE 1 = ENABLE
8	0x0	DISPLAY_INT_C0MASK: 0 = DISABLE 1 = ENABLE
6	0x0	GR3D_INT_C0MASK: 0 = DISABLE 1 = ENABLE
5	0x0	GR2D_INT_C0MASK: 0 = DISABLE 1 = ENABLE
4	0x0	ISP_INT_C0MASK: 0 = DISABLE 1 = ENABLE
3	0x0	EPP_INT_C0MASK: 0 = DISABLE 1 = ENABLE
2	0x0	VI_INT_C0MASK: 0 = DISABLE 1 = ENABLE
0	0x0	HOST_INT_C0MASK: 0 = DISABLE 1 = ENABLE

13.9.4 HOST1X_SYNC_INTC1MASK_0

INTC1MASK - Interrupt Mask for CPU1

Contains the interrupt mask bits for all of the client modules. If the INT_MASK is ENABLED for a particular module, the modules INT bit from INTSTATUS contributes to the CPU1's interrupt signal.

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxx00xx0xxx0x000x00000x0)

Bit	Reset	Description
20	0x0	TSEC_INT_C1MASK: 0 = DISABLE 1 = ENABLE
19	0x0	MSENC_INT_C1MASK: 0 = DISABLE 1 = ENABLE
16	0x0	DSIB_INT_C1MASK: 0 = DISABLE 1 = ENABLE
12	0x0	DSI_INT_C1MASK: 0 = DISABLE 1 = ENABLE
10	0x0	HDMI_INT_C1MASK: 0 = DISABLE 1 = ENABLE
9	0x0	DISPLAYB_INT_C1MASK: 0 = DISABLE 1 = ENABLE
8	0x0	DISPLAY_INT_C1MASK: 0 = DISABLE 1 = ENABLE
6	0x0	GR3D_INT_C1MASK: 0 = DISABLE 1 = ENABLE
5	0x0	GR2D_INT_C1MASK: 0 = DISABLE 1 = ENABLE
4	0x0	ISP_INT_C1MASK: 0 = DISABLE 1 = ENABLE
3	0x0	EPP_INT_C1MASK: 0 = DISABLE 1 = ENABLE
2	0x0	VI_INT_C1MASK: 0 = DISABLE 1 = ENABLE
0	0x0	HOST_INT_C1MASK: 0 = DISABLE 1 = ENABLE

13.9.5 HOST1X_SYNC_HINTSTATUS_0

Host Interrupt Status

Contains the interrupt status for the various host interrupts. The status is sticky and is PENDING until cleared (write 1's to INTSTATUS to clear).

Offset: 0x20 │ Read/Write: R/W │ Reset: 0x00000000 (0bx00xxxxxxxxx0000xxxxx00000000000)

Bit	R/W	Reset	Description
31	RO	X	HINTSTATUS_EXT_INT: Additional interrupts pending in the HINSTATUS_EXT register 0 = NOT_PENDING 1 = PENDING
30	RW	0x0	TIMER_INTP: Timer Interrupt from the Protected Channel 0 = NOT_PENDING 1 = PENDING
29	RW	0x0	CSW_HOST1XW2MC_INT: Host write client FIFO has filled up
19	RW	0x0	RDMA_DATABUF_THOLD_INT0: Read DMA data FIFO in port0 reached high level watermark 0 = NOT_PENDING 1 = PENDING
18	RW	0x0	RDMA_BUF_THOLD_INT0: Buffer threshold reached in read DMA port0 0 = NOT_PENDING 1 = PENDING
17	RW	0x0	RDMA_BUF_OFLOW_INT0: Buffer overflow in read DMA port0 0 = NOT_PENDING 1 = PENDING
16	RW	0x0	RDMA_INVAL_CLREQ_INT: Invalid client request to read DMA 0 = NOT_PENDING 1 = PENDING
10	RW	0x0	XBAR_TSEC_TIMEOUT_ID: ID of the CPU which timed out and updated the timeout address 0 = XBAR 1 = TSEC
9	RW	0x0	GR3D_ACTMON_INTR: MC_CPU Interrupt status. 1 = Interrupt detected 0 = Interrupt not detected 0 = NOINTR 1 = INTR
8	RW	0x0	WAIT_INT8: WAIT has completed on channel 8 0 = NOT_PENDING 1 = PENDING
7	RW	0x0	WAIT_INT7: WAIT has completed on channel 7 0 = NOT_PENDING 1 = PENDING
6	RW	0x0	WAIT_INT6: WAIT has completed on channel 6 0 = NOT_PENDING 1 = PENDING
5	RW	0x0	WAIT_INT5: WAIT has completed on channel 5 0 = NOT_PENDING 1 = PENDING
4	RW	0x0	WAIT_INT4: WAIT has completed on channel 4 0 = NOT_PENDING 1 = PENDING

Bit	R/W	Reset	Description
3	RW	0x0	WAIT_INT3: WAIT has completed on channel 3 0 = NOT_PENDING 1 = PENDING
2	RW	0x0	WAIT_INT2: WAIT has completed on channel 2 0 = NOT_PENDING 1 = PENDING
1	RW	0x0	WAIT_INT1: WAIT has completed on channel 1 0 = NOT_PENDING 1 = PENDING
0	RW	0x0	WAIT_INT0: WAIT has completed on channel 0 0 = NOT_PENDING 1 = PENDING

13.9.6 HOST1X_SYNC_HINTMASK_0

Host Interrupt Mask

Contains the interrupt mask bits for all of the host interrupts. If the INT_MASK is ENABLED for a particular module, the modules INT bit from INTSTATUS contributes to the global interrupt signal.

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x00000000 (0b000xxxxxxxxx0000xxx0xxx000000000)

Bit	Reset	Description
31	0x0	HINTSTATUS_EXT_INTMASK: 0 = DISABLE 1 = ENABLE
30	0x0	TIMER_INTMASKP: Timer Interrupt Mask for the Protected Channel 0 = DISABLE 1 = ENABLE
29	0x0	CSW_HOST1XW2MC_INTMASK: 0 = DISABLE 1 = ENABLE
19	0x0	RDMA_DATABUF_THOLD_INTMASK0: 0 = DISABLE 1 = ENABLE
18	0x0	RDMA_BUF_THOLD_INTMASK0: 0 = DISABLE 1 = ENABLE
17	0x0	RDMA_BUF_OFLOW_INTMASK0: 0 = DISABLE 1 = ENABLE
16	0x0	RDMA_INVAL_CLREQ_INTMASK: 0 = DISABLE 1 = ENABLE
12	0x0	GR3D_INTRMASK: MC_CPU Interrupt mask. 1 = Interrupt enabled 0 = Interrupt not enabled 0 = NOINTR 1 = INTR
8	0x0	WAIT_INTMASK8: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x0	WAIT_INTMASK7: 0 = DISABLE 1 = ENABLE
6	0x0	WAIT_INTMASK6: 0 = DISABLE 1 = ENABLE
5	0x0	WAIT_INTMASK5: 0 = DISABLE 1 = ENABLE
4	0x0	WAIT_INTMASK4: 0 = DISABLE 1 = ENABLE
3	0x0	WAIT_INTMASK3: 0 = DISABLE 1 = ENABLE
2	0x0	WAIT_INTMASK2: 0 = DISABLE 1 = ENABLE
1	0x0	WAIT_INTMASK1: 0 = DISABLE 1 = ENABLE
0	0x0	WAIT_INTMASK0: 0 = DISABLE 1 = ENABLE

13.9.7 HOST1X_SYNC_HINTSTATUS_EXT_0

Extended Host Interrupt Status

Contains additional interrupt status bits that did not fit in the HINTSTATUS register. When any of these bits is set, the HINTSTATUS_EXT_INT bit will also be set in the HINSTATUS register. Each status bit is sticky and PENDING until cleared (write 1's to HINTSTATUS_EXT to clear).

Offset: 0x28 │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
31	0x0	IP_WRITE_INT: Write transaction timeout occurred after IP_BUSY_TIMEOUT cycles. Offending address in IP_WRITE_TIMEOUT_ADDR. 0 = NOT_PENDING 1 = PENDING
30	0x0	IP_READ_INT: Read transaction timeout occurred after IP_BUSY_TIMEOUT cycles. Offending address in IP_READ_TIMEOUT_ADDR. 0 = NOT_PENDING 1 = PENDING
8	0x0	CMDPP_ILLEGAL_OPCODE_INT8: CMDPP8 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
7	0x0	CMDPP_ILLEGAL_OPCODE_INT7: CMDPP7 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
6	0x0	CMDPP_ILLEGAL_OPCODE_INT6: CMDPP6 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING

Bit	Reset	Description
5	0x0	CMDPP_ILLEGAL_OPCODE_INT5: CMDPP5 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
4	0x0	CMDPP_ILLEGAL_OPCODE_INT4: CMDPP4 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
3	0x0	CMDPP_ILLEGAL_OPCODE_INT3: CMDPP3 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
2	0x0	CMDPP_ILLEGAL_OPCODE_INT2: CMDPP2 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
1	0x0	CMDPP_ILLEGAL_OPCODE_INT1: CMDPP1 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING
0	0x0	CMDPP_ILLEGAL_OPCODE_INT0: CMDPP0 has seen an illegal opcode. 0 = NOT_PENDING 1 = PENDING

13.9.8 HOST1X_SYNC_HINTMASK_EXT_0

Offset: 0x2c │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
31	0x0	IP_WRITE_INTMASK: 0 = DISABLE 1 = ENABLE
30	0x0	IP_READ_INTMASK: 0 = DISABLE 1 = ENABLE
8	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK8: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
7	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK7: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
6	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK6: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
5	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK5: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
4	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK4: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
3	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK3: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
2	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK2: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE
1	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK1: See CMDPP_ILLEGAL_OPCODE_INTMASK0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	CMDPP_ILLEGAL_OPCODE_INTMASK0: Mask CMDPP_ILLEGAL_OPCODE_INT0 interrupt bit. 0 = DISABLE 1 = ENABLE

13.9.9 HOST1X_SYNC_SYNCPT_THRESH_CPU0_INT_STATUS_0

Syncpt Threshold Interrupt Status

Two registers -- one for the CPU complex (CPU0) and one for the AVP for testing purposes (CPU1). Status is sticky and is PENDING until cleared. An interrupt routed to cpuN occurs when

(SYNCPT[indx] >= SYNCPT_INT_THRESH[indx]) && (SYNCPT_THRESH_INT_MASK[indx] == cpuN) (compare takes into account wrap). Write 1's to clear.

Offset: 0x40 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_31: 0 = NOT_PENDING 1 = PENDING
30	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_30: 0 = NOT_PENDING 1 = PENDING
29	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_29: 0 = NOT_PENDING 1 = PENDING
28	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_28: 0 = NOT_PENDING 1 = PENDING
27	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_27: 0 = NOT_PENDING 1 = PENDING
26	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_26: 0 = NOT_PENDING 1 = PENDING
25	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_25: 0 = NOT_PENDING 1 = PENDING
24	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_24: 0 = NOT_PENDING 1 = PENDING
23	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_23: 0 = NOT_PENDING 1 = PENDING
22	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_22: 0 = NOT_PENDING 1 = PENDING
21	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_21: 0 = NOT_PENDING 1 = PENDING
20	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_20: 0 = NOT_PENDING 1 = PENDING

Bit	Reset	Description
19	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_19: 0 = NOT_PENDING 1 = PENDING
18	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_18: 0 = NOT_PENDING 1 = PENDING
17	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_17: 0 = NOT_PENDING 1 = PENDING
16	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_16: 0 = NOT_PENDING 1 = PENDING
15	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_15: 0 = NOT_PENDING 1 = PENDING
14	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_14: 0 = NOT_PENDING 1 = PENDING
13	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_13: 0 = NOT_PENDING 1 = PENDING
12	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_12: 0 = NOT_PENDING 1 = PENDING
11	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_11: 0 = NOT_PENDING 1 = PENDING
10	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_10: 0 = NOT_PENDING 1 = PENDING
9	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_9: 0 = NOT_PENDING 1 = PENDING
8	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_8: 0 = NOT_PENDING 1 = PENDING
7	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_7: 0 = NOT_PENDING 1 = PENDING
6	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_6: 0 = NOT_PENDING 1 = PENDING
5	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_5: 0 = NOT_PENDING 1 = PENDING
4	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_4: 0 = NOT_PENDING 1 = PENDING
3	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_3: 0 = NOT_PENDING 1 = PENDING
2	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_2: 0 = NOT_PENDING 1 = PENDING

Bit	Reset	Description
1	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_1: 0 = NOT_PENDING 1 = PENDING
0	0x0	SYNCPT_THRESH_CPU0_INT_STATUS_0: 0 = NOT_PENDING 1 = PENDING

13.9.10 HOST1X_SYNC_SYNCPT_THRESH_CPU1_INT_STATUS_0

Offset: 0x48 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_31: 0 = NOT_PENDING 1 = PENDING
30	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_30: 0 = NOT_PENDING 1 = PENDING
29	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_29: 0 = NOT_PENDING 1 = PENDING
28	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_28: 0 = NOT_PENDING 1 = PENDING
27	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_27: 0 = NOT_PENDING 1 = PENDING
26	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_26: 0 = NOT_PENDING 1 = PENDING
25	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_25: 0 = NOT_PENDING 1 = PENDING
24	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_24: 0 = NOT_PENDING 1 = PENDING
23	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_23: 0 = NOT_PENDING 1 = PENDING
22	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_22: 0 = NOT_PENDING 1 = PENDING
21	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_21: 0 = NOT_PENDING 1 = PENDING
20	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_20: 0 = NOT_PENDING 1 = PENDING
19	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_19: 0 = NOT_PENDING 1 = PENDING
18	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_18: 0 = NOT_PENDING 1 = PENDING

Bit	Reset	Description
17	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_17: 0 = NOT_PENDING 1 = PENDING
16	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_16: 0 = NOT_PENDING 1 = PENDING
15	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_15: 0 = NOT_PENDING 1 = PENDING
14	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_14: 0 = NOT_PENDING 1 = PENDING
13	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_13: 0 = NOT_PENDING 1 = PENDING
12	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_12: 0 = NOT_PENDING 1 = PENDING
11	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_11: 0 = NOT_PENDING 1 = PENDING
10	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_10: 0 = NOT_PENDING 1 = PENDING
9	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_9: 0 = NOT_PENDING 1 = PENDING
8	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_8: 0 = NOT_PENDING 1 = PENDING
7	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_7: 0 = NOT_PENDING 1 = PENDING
6	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_6: 0 = NOT_PENDING 1 = PENDING
5	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_5: 0 = NOT_PENDING 1 = PENDING
4	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_4: 0 = NOT_PENDING 1 = PENDING
3	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_3: 0 = NOT_PENDING 1 = PENDING
2	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_2: 0 = NOT_PENDING 1 = PENDING
1	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_1: 0 = NOT_PENDING 1 = PENDING
0	0x0	SYNCPT_THRESH_CPU1_INT_STATUS_0: 0 = NOT_PENDING 1 = PENDING

13.9.11 HOST1X_SYNC_SYNCPT_THRESH_INT_MASK_0_0

Syncpt Threshold Interrupt Mask

If set to ENABLE_CPU0, interrupts are routed to the CPU complex.

If set to ENABLE_CPU1, interrupts are routed to the AVP.

Offset: 0x50 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:30	0x0	SYNCPT_THRESH_INT_MASK_15: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
29:28	0x0	SYNCPT_THRESH_INT_MASK_14: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
27:26	0x0	SYNCPT_THRESH_INT_MASK_13: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
25:24	0x0	SYNCPT_THRESH_INT_MASK_12: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
23:22	0x0	SYNCPT_THRESH_INT_MASK_11: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
21:20	0x0	SYNCPT_THRESH_INT_MASK_10: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
19:18	0x0	SYNCPT_THRESH_INT_MASK_9: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
17:16	0x0	SYNCPT_THRESH_INT_MASK_8: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
15:14	0x0	SYNCPT_THRESH_INT_MASK_7: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
13:12	0x0	SYNCPT_THRESH_INT_MASK_6: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
11:10	0x0	SYNCPT_THRESH_INT_MASK_5: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
9:8	0x0	SYNCPT_THRESH_INT_MASK_4: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1

Bit	Reset	Description
7:6	0x0	SYNCPT_THRESH_INT_MASK_3: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
5:4	0x0	SYNCPT_THRESH_INT_MASK_2: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
3:2	0x0	SYNCPT_THRESH_INT_MASK_1: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
1:0	0x0	SYNCPT_THRESH_INT_MASK_0: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1

13.9.12 HOST1X_SYNC_SYNCPT_THRESH_INT_MASK_1_0

Offset: 0x54 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:30	0x0	SYNCPT_THRESH_INT_MASK_31: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
29:28	0x0	SYNCPT_THRESH_INT_MASK_30: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
27:26	0x0	SYNCPT_THRESH_INT_MASK_29: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
25:24	0x0	SYNCPT_THRESH_INT_MASK_28: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
23:22	0x0	SYNCPT_THRESH_INT_MASK_27: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
21:20	0x0	SYNCPT_THRESH_INT_MASK_26: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
19:18	0x0	SYNCPT_THRESH_INT_MASK_25: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
17:16	0x0	SYNCPT_THRESH_INT_MASK_24: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
15:14	0x0	SYNCPT_THRESH_INT_MASK_23: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1

Bit	Reset	Description
13:12	0x0	SYNCPT_THRESH_INT_MASK_22: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
11:10	0x0	SYNCPT_THRESH_INT_MASK_21: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
9:8	0x0	SYNCPT_THRESH_INT_MASK_20: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
7:6	0x0	SYNCPT_THRESH_INT_MASK_19: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
5:4	0x0	SYNCPT_THRESH_INT_MASK_18: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
3:2	0x0	SYNCPT_THRESH_INT_MASK_17: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1
1:0	0x0	SYNCPT_THRESH_INT_MASK_16: 0 = DISABLE 1 = ENABLE_CPU0 2 = ENABLE_CPU1

13.9.13 HOST1X_SYNC_SYNCPT_THRESH_INT_DISABLE_0

Write-only registers to change the state of the SYNCPT_THRESH_INT_MASK fields. Write a 1 to the Nth position to change syncpt[N]'s threshold interrupt mask. This register provides an alternative way to change the mask, which is multiprocessor safe.

Offset: 0x60 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SYNCPT_THRESH_INT_DISABLE_31: 0 = NO_CHANGE 1 = DISABLE
30	0x0	SYNCPT_THRESH_INT_DISABLE_30: 0 = NO_CHANGE 1 = DISABLE
29	0x0	SYNCPT_THRESH_INT_DISABLE_29: 0 = NO_CHANGE 1 = DISABLE
28	0x0	SYNCPT_THRESH_INT_DISABLE_28: 0 = NO_CHANGE 1 = DISABLE
27	0x0	SYNCPT_THRESH_INT_DISABLE_27: 0 = NO_CHANGE 1 = DISABLE
26	0x0	SYNCPT_THRESH_INT_DISABLE_26: 0 = NO_CHANGE 1 = DISABLE

Bit	Reset	Description
25	0x0	SYNCPT_THRESH_INT_DISABLE_25: 0 = NO_CHANGE 1 = DISABLE
24	0x0	SYNCPT_THRESH_INT_DISABLE_24: 0 = NO_CHANGE 1 = DISABLE
23	0x0	SYNCPT_THRESH_INT_DISABLE_23: 0 = NO_CHANGE 1 = DISABLE
22	0x0	SYNCPT_THRESH_INT_DISABLE_22: 0 = NO_CHANGE 1 = DISABLE
21	0x0	SYNCPT_THRESH_INT_DISABLE_21: 0 = NO_CHANGE 1 = DISABLE
20	0x0	SYNCPT_THRESH_INT_DISABLE_20: 0 = NO_CHANGE 1 = DISABLE
19	0x0	SYNCPT_THRESH_INT_DISABLE_19: 0 = NO_CHANGE 1 = DISABLE
18	0x0	SYNCPT_THRESH_INT_DISABLE_18: 0 = NO_CHANGE 1 = DISABLE
17	0x0	SYNCPT_THRESH_INT_DISABLE_17: 0 = NO_CHANGE 1 = DISABLE
16	0x0	SYNCPT_THRESH_INT_DISABLE_16: 0 = NO_CHANGE 1 = DISABLE
15	0x0	SYNCPT_THRESH_INT_DISABLE_15: 0 = NO_CHANGE 1 = DISABLE
14	0x0	SYNCPT_THRESH_INT_DISABLE_14: 0 = NO_CHANGE 1 = DISABLE
13	0x0	SYNCPT_THRESH_INT_DISABLE_13: 0 = NO_CHANGE 1 = DISABLE
12	0x0	SYNCPT_THRESH_INT_DISABLE_12: 0 = NO_CHANGE 1 = DISABLE
11	0x0	SYNCPT_THRESH_INT_DISABLE_11: 0 = NO_CHANGE 1 = DISABLE
10	0x0	SYNCPT_THRESH_INT_DISABLE_10: 0 = NO_CHANGE 1 = DISABLE
9	0x0	SYNCPT_THRESH_INT_DISABLE_9: 0 = NO_CHANGE 1 = DISABLE
8	0x0	SYNCPT_THRESH_INT_DISABLE_8: 0 = NO_CHANGE 1 = DISABLE

Bit	Reset	Description
7	0x0	SYNCPT_THRESH_INT_DISABLE_7: 0 = NO_CHANGE 1 = DISABLE
6	0x0	SYNCPT_THRESH_INT_DISABLE_6: 0 = NO_CHANGE 1 = DISABLE
5	0x0	SYNCPT_THRESH_INT_DISABLE_5: 0 = NO_CHANGE 1 = DISABLE
4	0x0	SYNCPT_THRESH_INT_DISABLE_4: 0 = NO_CHANGE 1 = DISABLE
3	0x0	SYNCPT_THRESH_INT_DISABLE_3: 0 = NO_CHANGE 1 = DISABLE
2	0x0	SYNCPT_THRESH_INT_DISABLE_2: 0 = NO_CHANGE 1 = DISABLE
1	0x0	SYNCPT_THRESH_INT_DISABLE_1: 0 = NO_CHANGE 1 = DISABLE
0	0x0	SYNCPT_THRESH_INT_DISABLE_0: 0 = NO_CHANGE 1 = DISABLE

13.9.14 HOST1X_SYNC_SYNCPT_THRESH_INT_ENABLE_CPU0_0

Offset: 0x68 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_31: 0 = NO_CHANGE 1 = ENABLE_CPU0
30	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_30: 0 = NO_CHANGE 1 = ENABLE_CPU0
29	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_29: 0 = NO_CHANGE 1 = ENABLE_CPU0
28	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_28: 0 = NO_CHANGE 1 = ENABLE_CPU0
27	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_27: 0 = NO_CHANGE 1 = ENABLE_CPU0
26	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_26: 0 = NO_CHANGE 1 = ENABLE_CPU0
25	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_25: 0 = NO_CHANGE 1 = ENABLE_CPU0
24	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_24: 0 = NO_CHANGE 1 = ENABLE_CPU0

Bit	Reset	Description
23	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_23: 0 = NO_CHANGE 1 = ENABLE_CPU0
22	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_22: 0 = NO_CHANGE 1 = ENABLE_CPU0
21	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_21: 0 = NO_CHANGE 1 = ENABLE_CPU0
20	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_20: 0 = NO_CHANGE 1 = ENABLE_CPU0
19	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_19: 0 = NO_CHANGE 1 = ENABLE_CPU0
18	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_18: 0 = NO_CHANGE 1 = ENABLE_CPU0
17	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_17: 0 = NO_CHANGE 1 = ENABLE_CPU0
16	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_16: 0 = NO_CHANGE 1 = ENABLE_CPU0
15	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_15: 0 = NO_CHANGE 1 = ENABLE_CPU0
14	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_14: 0 = NO_CHANGE 1 = ENABLE_CPU0
13	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_13: 0 = NO_CHANGE 1 = ENABLE_CPU0
12	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_12: 0 = NO_CHANGE 1 = ENABLE_CPU0
11	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_11: 0 = NO_CHANGE 1 = ENABLE_CPU0
10	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_10: 0 = NO_CHANGE 1 = ENABLE_CPU0
9	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_9: 0 = NO_CHANGE 1 = ENABLE_CPU0
8	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_8: 0 = NO_CHANGE 1 = ENABLE_CPU0
7	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_7: 0 = NO_CHANGE 1 = ENABLE_CPU0
6	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_6: 0 = NO_CHANGE 1 = ENABLE_CPU0

Bit	Reset	Description
5	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_5: 0 = NO_CHANGE 1 = ENABLE_CPU0
4	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_4: 0 = NO_CHANGE 1 = ENABLE_CPU0
3	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_3: 0 = NO_CHANGE 1 = ENABLE_CPU0
2	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_2: 0 = NO_CHANGE 1 = ENABLE_CPU0
1	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_1: 0 = NO_CHANGE 1 = ENABLE_CPU0
0	0x0	SYNCPT_THRESH_INT_ENABLE_CPU0_0: 0 = NO_CHANGE 1 = ENABLE_CPU0

13.9.15 HOST1X_SYNC_SYNCPT_THRESH_INT_ENABLE_CPU1_0

Offset: 0x70 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_31: 0 = NO_CHANGE 1 = ENABLE_CPU1
30	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_30: 0 = NO_CHANGE 1 = ENABLE_CPU1
29	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_29: 0 = NO_CHANGE 1 = ENABLE_CPU1
28	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_28: 0 = NO_CHANGE 1 = ENABLE_CPU1
27	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_27: 0 = NO_CHANGE 1 = ENABLE_CPU1
26	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_26: 0 = NO_CHANGE 1 = ENABLE_CPU1
25	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_25: 0 = NO_CHANGE 1 = ENABLE_CPU1
24	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_24: 0 = NO_CHANGE 1 = ENABLE_CPU1
23	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_23: 0 = NO_CHANGE 1 = ENABLE_CPU1
22	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_22: 0 = NO_CHANGE 1 = ENABLE_CPU1

Bit	Reset	Description
21	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_21: 0 = NO_CHANGE 1 = ENABLE_CPU1
20	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_20: 0 = NO_CHANGE 1 = ENABLE_CPU1
19	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_19: 0 = NO_CHANGE 1 = ENABLE_CPU1
18	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_18: 0 = NO_CHANGE 1 = ENABLE_CPU1
17	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_17: 0 = NO_CHANGE 1 = ENABLE_CPU1
16	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_16: 0 = NO_CHANGE 1 = ENABLE_CPU1
15	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_15: 0 = NO_CHANGE 1 = ENABLE_CPU1
14	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_14: 0 = NO_CHANGE 1 = ENABLE_CPU1
13	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_13: 0 = NO_CHANGE 1 = ENABLE_CPU1
12	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_12: 0 = NO_CHANGE 1 = ENABLE_CPU1
11	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_11: 0 = NO_CHANGE 1 = ENABLE_CPU1
10	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_10: 0 = NO_CHANGE 1 = ENABLE_CPU1
9	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_9: 0 = NO_CHANGE 1 = ENABLE_CPU1
8	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_8: 0 = NO_CHANGE 1 = ENABLE_CPU1
7	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_7: 0 = NO_CHANGE 1 = ENABLE_CPU1
6	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_6: 0 = NO_CHANGE 1 = ENABLE_CPU1
5	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_5: 0 = NO_CHANGE 1 = ENABLE_CPU1
4	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_4: 0 = NO_CHANGE 1 = ENABLE_CPU1

Bit	Reset	Description
3	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_3: 0 = NO_CHANGE 1 = ENABLE_CPU1
2	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_2: 0 = NO_CHANGE 1 = ENABLE_CPU1
1	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_1: 0 = NO_CHANGE 1 = ENABLE_CPU1
0	0x0	SYNCPT_THRESH_INT_ENABLE_CPU1_0: 0 = NO_CHANGE 1 = ENABLE_CPU1

13.9.16 HOST1X_SYNC_CF0_SETUP_0

Channel Processor and FIFO Setup, Control, Status

Note: Command FIFO setup registers set up the various regions of the command FIFO. SHOULD ONLY BE CHANGED WHEN THE FIFO IS EMPTY. These are initialized to 8 equal-sized FIFOs.

Offset: 0x80 │ Read/Write: R/W │ Reset: 0x003f0000 (0bxxxxxx0000111111xxxxxx0000000000)

Bit	Reset	Description
25:16	0x3f	CF0_LIMIT: Channel 0 FIFO limit (highest address)
9:0	0x0	CF0_BASE: Channel 0 FIFO base

13.9.17 HOST1X_SYNC_CF1_SETUP_0

Offset: 0x84 │ Read/Write: R/W │ Reset: 0x007f0040 (0bxxxxxx0001111111xxxxxx0001000000)

Bit	Reset	Description
25:16	0x7f	CF1_LIMIT: Channel 1 FIFO limit (highest address)
9:0	0x40	CF1_BASE: Channel 1 FIFO base

13.9.18 HOST1X_SYNC_CF2_SETUP_0

Offset: 0x88 │ Read/Write: R/W │ Reset: 0x00bf0080 (0bxxxxxx0010111111xxxxxx0010000000)

Bit	Reset	Description
25:16	0xbf	CF2_LIMIT: Channel 2 FIFO limit (highest address)
9:0	0x80	CF2_BASE: Channel 2 FIFO base

13.9.19 HOST1X_SYNC_CF3_SETUP_0

Offset: 0x8c │ Read/Write: R/W │ Reset: 0x00ff00c0 (0bxxxxxx0011111111xxxxxx0011000000)

Bit	Reset	Description
25:16	0xff	CF3_LIMIT: Channel 3 FIFO limit (highest address)
9:0	0xc0	CF3_BASE: Channel 3 FIFO base

13.9.20 HOST1X_SYNC_CF4_SETUP_0

Offset: 0x90 | Read/Write: R/W | Reset: 0x013f0100 (0bxxxxxx0100111111xxxxxx0100000000)

Bit	Reset	Description
25:16	0x13f	CF4_LIMIT: Channel 4 FIFO limit (highest address)
9:0	0x100	CF4_BASE: Channel 4 FIFO base

13.9.21 HOST1X_SYNC_CF5_SETUP_0

Offset: 0x94 | Read/Write: R/W | Reset: 0x017f0140 (0bxxxxxx0101111111xxxxxx0101000000)

Bit	Reset	Description
25:16	0x17f	CF5_LIMIT: Channel 5 FIFO limit (highest address)
9:0	0x140	CF5_BASE: Channel 5 FIFO base

13.9.22 HOST1X_SYNC_CF6_SETUP_0

Offset: 0x98 | Read/Write: R/W | Reset: 0x01bf0180 (0bxxxxxx0110111111xxxxxx0110000000)

Bit	Reset	Description
25:16	0x1bf	CF6_LIMIT: Channel 6 FIFO limit (highest address)
9:0	0x180	CF6_BASE: Channel 6 FIFO base

13.9.23 HOST1X_SYNC_CF7_SETUP_0

Offset: 0x9c | Read/Write: R/W | Reset: 0x01ff01c0 (0bxxxxxx0111111111xxxxxx0111000000)

Bit	Reset	Description
25:16	0x1ff	CF7_LIMIT: Channel 7 FIFO limit (highest address)
9:0	0x1c0	CF7_BASE: Channel 7 FIFO base

13.9.24 HOST1X_SYNC_CF_SETUPDONE_0

Write to this register to trigger an update of the FIFO's pointers. **ONLY DO THIS WHEN THE FIFO IS EMPTY.**

Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	CF_SETUPDONE: Dummy bit

13.9.25 HOST1X_SYNC_CMDPROC_CTRL_0

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0xx00x)

Bit	Reset	Description
5	0x0	INTFC_CLKEN_OVR
2	0x0	GATHER_PARSE_DISABLED
1	0x0	DROP_ILLEGAL_OPCODES

13.9.26 HOST1X_SYNC_CMDPROC_STAT_0

Offset: 0xa8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8:0	X	ILLEGAL_OPCODE

13.9.27 HOST1X_SYNC_CMDPROC_STOP_0

CH*_CMDPROC_STOP stops issuing commands from the command FIFO. This is useful to stop other channels when a channel teardown is needed to prevent unwanted traffic from happening at the same time.

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	CH8_CMDPROC_STOP: 0 = RUN 1 = STOP
7	0x0	CH7_CMDPROC_STOP: 0 = RUN 1 = STOP
6	0x0	CH6_CMDPROC_STOP: 0 = RUN 1 = STOP
5	0x0	CH5_CMDPROC_STOP: 0 = RUN 1 = STOP
4	0x0	CH4_CMDPROC_STOP: 0 = RUN 1 = STOP
3	0x0	CH3_CMDPROC_STOP: 0 = RUN 1 = STOP
2	0x0	CH2_CMDPROC_STOP: 0 = RUN 1 = STOP
1	0x0	CH1_CMDPROC_STOP: 0 = RUN 1 = STOP
0	0x0	CH0_CMDPROC_STOP: 0 = RUN 1 = STOP

13.9.28 HOST1X_SYNC_CH_TEARDOWN_0

Channel teardown register. Tells the hardware that a channel has gone away. Will reset that channel's command FIFO and release any locks it has in the arbiter. Will NOT reset that channel's output FIFO, which can be emptied by reading out all remaining entries.

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	CH8_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN

Bit	Reset	Description
7	X	CH7_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
6	X	CH6_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
5	X	CH5_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
4	X	CH4_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
3	X	CH3_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
2	X	CH2_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
1	X	CH1_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
0	X	CH0_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN

13.9.29 HOST1X_SYNC_MOD_TEARDOWN_0

Module teardown register. If a module is reset, the host needs to reset its state with respect to which channels own that module. Whenever a module is reset, the corresponding teardown bit in this register should be written. Module teardown will only work if the command FIFO and command processor are in a good state (the FIFO is empty of traffic for that channel and the command processor is at an opcode boundary). If an entire channel needs to be reset, the CH_TEARDOWN register should be used instead.

Offset: 0xb4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20	X	TSEC_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
19	X	MSENC_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
16	X	DSIB_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
12	X	DSI_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
10	X	HDMI_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
9	X	DISPLAYB_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN

Bit	Reset	Description
8	X	DISPLAY_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
6	X	GR3D_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
5	X	GR2D_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
4	X	ISP_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
3	X	EPP_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN
2	X	VI_TEARDOWN: 0 = NO_ACTION 1 = TEARDOWN

13.9.30 HOST1X_SYNC_CH0_STATUS_0

The channel status registers show each channel's working class. CHOUT_CLASS* holds the requested class in the case of a blocked context switch.

Offset: 0xb8 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CTXSW_PENDING0
25:16	X	CHOUT_CLASS0: Current or blocked (requested) class for channel 0

13.9.31 HOST1X_SYNC_CH1_STATUS_0

Offset: 0xbc │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CTXSW_PENDING1
25:16	X	CHOUT_CLASS1: Current or blocked (requested) class for channel 1

13.9.32 HOST1X_SYNC_CH2_STATUS_0

Offset: 0xc0 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CTXSW_PENDING2
25:16	X	CHOUT_CLASS2: Current or blocked (requested) class for channel 2

13.9.33 HOST1X_SYNC_CH3_STATUS_0

Offset: 0xc4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CTXSW_PENDING3
25:16	X	CHOUT_CLASS3: Current or blocked (requested) class for channel 3

13.9.34 HOST1X_SYNC_CH4_STATUS_0

Offset: 0xc8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CTXSW_PENDING4
25:16	X	CHOUT_CLASS4: Current or blocked (requested) class for channel 4

13.9.35 HOST1X_SYNC_CH5_STATUS_0

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

Bit	Reset	Description
31	X	CTXSW_PENDING5
25:16	X	CHOUT_CLASS5: Current or blocked (requested) class for channel 5

13.9.36 HOST1X_SYNC_CH6_STATUS_0

Offset: 0xd0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CTXSW_PENDING6
25:16	X	CHOUT_CLASS6: Current or blocked (requested) class for channel 6

13.9.37 HOST1X_SYNC_CH7_STATUS_0

Offset: 0xd4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CTXSW_PENDING7
25:16	X	CHOUT_CLASS7: Current or blocked (requested) class for channel 7

13.9.38 HOST1X_SYNC_DISPLAY_STATUS_0

The per-client status registers indicate which channel owns each client as well as the current working class for each client. This is useful for determining each client's state with regard to granting context switches. _CURRCL is the current class ID for that module.

Offset: 0xd8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DISPLAY_CURRCL

13.9.39 HOST1X_SYNC_DISPLAYB_STATUS_0

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

Bit	Reset	Description
25:16	X	DISPLAYB_CURRCL

13.9.40 HOST1X_SYNC_EPP_STATUS_0

Offset: 0xe0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	EPP_CURRCL

13.9.41 HOST1X_SYNC_GR3D_STATUS_0

Offset: 0xe4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	GR3D_CURRCL

13.9.42 HOST1X_SYNC_ISP_STATUS_0

Offset: 0xe8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	ISP_CURRCL

13.9.43 HOST1X_SYNC_DSI_STATUS_0

Offset: 0xf4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DSI_CURRCL

13.9.44 HOST1X_SYNC_HDMI_STATUS_0

Offset: 0xf8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	HDMI_CURRCL

13.9.45 HOST1X_SYNC_VI_STATUS_0

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

Bit	Reset	Description
25:16	X	VI_CURRCL

13.9.46 HOST1X_SYNC_GR2D_STATUS_0

Offset: 0x100 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	GR2D_CURRCL

13.9.47 HOST1X_SYNC_DSIB_STATUS_0

Offset: 0x104 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	DSIB_CURRCL

13.9.48 HOST1X_SYNC_MSENC_STATUS_0

Offset: 0x108 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	MSENC_CURRCL

13.9.49 HOST1X_SYNC_TSEC_STATUS_0

Offset: 0x10c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	TSEC_CURRCL

13.9.50 HOST1X_SYNC_DIRECT_MODULE_CONFIG_0

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x54000000 (0b01010100000000000000000000000xx)

Bit	Reset	Description
31:2	0x15000000	BASE

13.9.51 HOST1X_SYNC_USEC_CLK_0

Number of host clocks needed to make a microsecond. Used for the DELAY host method. For example, if the host clock is 250 MHz, this register should be programmed to 250. If the host clock is 150 MHz, this register should be programmed to 150.

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x0000015e (0bxxxxxxxxxxxxxxxxxxxx000101011110)

Bit	Reset	Description
11:0	0x15e	USEC_CLKS

13.9.52 HOST1X_SYNC_CTXSW_TIMEOUT_CFG_0

With channel ownership of modules removed, only generate CTXSW if a channel writes to a new class on a client. To keep from continually switching contexts if channels are addressing same client, keep a counter of commands issued to the client, and do not switch until the client has either received WAIT_CTXSW_CNT clocks before switching or the channel stops targeting it. So if 4 channels are currently in use with 2 channels targeting different classes on the same client, if the value were set to 6, it would guarantee 2 commands were sent from any channel to the client's current class.

Offset: 0x1a8 │ Read/Write: R/W │ Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00001111)

Bit	Reset	Description
7:0	0xf	WAIT_CTXSW_CNT: Number of cycles to wait

13.9.53 HOST1X_SYNC_INDREG_DMA_CTRL_0

Enable use of DMA engine to move data from indirect register interface to memory. ATTN_LVL controls flow between host and DMA. DMA will not start a transfer until the set number of slots are full:

- 00 = 1 slot
- 01 = 4 slots
- 10 = 8 slots

Offset: 0x1ac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000x0000)

Bit	Reset	Description
7	0x0	AHBDMA_ENABLE: Enable generation of request to DMA engine 0 = DISABLE 1 = ENABLE
6:5	0x0	AHBDMA_ATTN_LVL: Number of entries to receive before sending DMA request, 1, 4, or 8
3:0	0x0	AHBDMA_CHID: channel being used by indirect read for DMA (which chout FIFO to monitor)

13.9.54 HOST1X_SYNC_CHANNEL_PRIORITY_0

Set channel priority for dual ring arbitration (hi/lo). Used in arbitrating MLOCKs.

Offset: 0x1b0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	HIPRI_CH8: 0 = DISABLE 1 = ENABLE
7	0x0	HIPRI_CH7: 0 = DISABLE 1 = ENABLE
6	0x0	HIPRI_CH6: 0 = DISABLE 1 = ENABLE
5	0x0	HIPRI_CH5: 0 = DISABLE 1 = ENABLE
4	0x0	HIPRI_CH4: 0 = DISABLE 1 = ENABLE
3	0x0	HIPRI_CH3: 0 = DISABLE 1 = ENABLE
2	0x0	HIPRI_CH2: 0 = DISABLE 1 = ENABLE
1	0x0	HIPRI_CH1: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	HIPRI_CH0: 0 = DISABLE 1 = ENABLE

13.9.55 HOST1X_SYNC_CDMA_ASM_TIMEOUT_0

Timeout value determines how long the assembly logic will wait before it flushes data from the data FIFO to avoid head-of-line blocking.

Note: Do not use a value of zero -- causes immediate flushing so no forward progress is made.

Offset: 0x1b4 | Read/Write: R/W | Reset: 0x00000404 (0bxxxxxxxxxxxxxxxxxxxx00100xxx00100)

Bit	Reset	Description
12:8	0x4	CDMA_ASM_GFIFO_TIMEOUT
4:0	0x4	CDMA_ASM_DFIFO_TIMEOUT

13.9.56 HOST1X_SYNC_CDMA_MISC_0

CDMA_DELAY_PUT

Delay put_addr updates. This can potentially increase MC performance slightly by gathering multiple consecutive put_addr updates into 1 update, which reduces the amount of requests to the MC. In addition, it can be used to cover a race condition where a write to memory has not completed before CDMA starts fetching the data. But in this case, the counter has to be set to a high value, which *will* affect performance.

CDMA_SIMPLE_PREFETCH

Reduce per-channel requests to 1 every 3 cycles. This can reduce performance in 2 ways:

- The memory request FIFO will fill up slightly slower, but the impact will be minimal.
- It will cause the channel arbiter to switch to other channels when they have a request available, which will reduce locality. On the plus side, it has much less nasty corner cases.

CDMA_PUT_SYNC_DISABLE

CDMA_EN_STATS

Enable statistics counters.

- - Count the number of 128-bit words fetched by CDMA
- - Count the number of 128-bit words thrown away by CDMA

This allows us to get an idea how efficient the CDMA really is when there are multiple push-buffers active at the same time.

Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0x000xx000000)

Bit	Reset	Description
12	0x0	CDMA_EN_STATS
10	0x0	CDMA_CLKEN_OVR
9	0x0	CDMA_PUT_SYNC_DISABLE
8	0x0	CDMA_SIMPLE_PREFETCH
5:0	0x0	CDMA_DELAY_PUT

13.9.57 HOST1X_SYNC_IP_BUSY_TIMEOUT_0

Offset: 0x1bc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	IP_BUSY_TIMEOUT: Number of busy cycles before requesting a retry. 0 = disabled

13.9.58 HOST1X_SYNC_IP_READ_TIMEOUT_ADDR_0

Offset: 0x1c0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	IP_READ_TIMEOUT_ADDR: Address of transaction that caused an AXI read timeout

13.9.59 HOST1X_SYNC_IP_WRITE_TIMEOUT_ADDR_0

Offset: 0x1c4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:2	X	IP_WRITE_TIMEOUT_ADDR: Address of transaction that caused an AXI write timeout

13.9.60 HOST1X_SYNC_MCCIF_THCTRL_0

Memory write client FIFO status. Reads out the available 128-bit entries. If writing through the buffered frame buffer write region, writes are accumulated up to 128 bits before being flushed, so 10 entries can mean up to 40 writes. Memory client high-priority threshold control register. Sets the threshold of when the client becomes high priority.

Offset: 0x1d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx000000xxxxxxxx)

Bit	R/W	Reset	Description
13:8	RW	0x0	CSW_HOST1XW2MC_HPTH
5:0	RO	X	CSW_HOST1XW_FIFOSTAT

13.9.61 HOST1X_SYNC_HC_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register

Note: This FIFO timing aspects of this register are no longer supported, but are retained for software compatibility.

The clock enable fields of this register control the second-level clock gating for the MC side of the MCCIF. A '1' written to the override field will result in that clock reverting to the legacy mode of operation (where the msc*clk is on whenever the client clock is enabled.

Offset: 0x1dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx00xxxxxxxxxxx0000)

Bit	Reset	Description
17	0x0	HC_RCLK_OVERRIDE
16	0x0	HC_WCLK_OVERRIDE
3	DISABLE	HC_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	DISABLE	HC_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	HC_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	HC_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

13.9.62 HOST1X_SYNC_TIMEOUT_WCOAL_HC_0

Write Coalescing Time-Out Register

Note: Write coalescing is no longer supported by the MCCIF clients. Registers are retained for software compatibility but are not used by the hardware.

Offset: 0x1e0 │ Read/Write: R/W │ Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	HOST1XW_WCOAL_TMVAL

13.9.63 HOST1X_SYNC_HWLOCK0_0

HWLOCK REGISTERS

Hardware lock registers. These registers are general-purpose and allow software to do an atomic operation to gain ownership of a resource. These registers read back '1' if the reader has obtained ownership and '0' if not. This is done by initializing the register to '1'. The act of reading it resets it to '0'. Software must then write '1' back to it when it no longer needs the resource.

Offset: 0x280 │ Read/Write: RO │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	HWLOCK0

13.9.64 HOST1X_SYNC_HWLOCK1_0

Offset: 0x284 │ Read/Write: RO │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	HWLOCK1

13.9.65 HOST1X_SYNC_HWLOCK2_0

Offset: 0x288 │ Read/Write: RO │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	HWLOCK2

13.9.66 HOST1X_SYNC_HWLOCK3_0

Offset: 0x28c | Read/Write: RO | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	HWLOCK3

13.9.67 HOST1X_SYNC_HWLOCK4_0

Offset: 0x290 | Read/Write: RO | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	HWLOCK4

13.9.68 HOST1X_SYNC_HWLOCK5_0

Offset: 0x294 | Read/Write: RO | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	HWLOCK5

13.9.69 HOST1X_SYNC_HWLOCK6_0

Offset: 0x298 | Read/Write: RO | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	HWLOCK6

13.9.70 HOST1X_SYNC_HWLOCK7_0

Offset: 0x29c | Read/Write: RO | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	HWLOCK7

13.9.71 HOST1X_SYNC_HWLOCK8_0

Offset: 0x2a0 | Read/Write: RO | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	HWLOCK8

13.9.72 HOST1X_SYNC_MLOCK_0_0

MLOCK Registers

MLOCK is 1 whenever someone holds the lock, and is 0 otherwise. MLOCK is normally set and cleared using the host methods ACQUIRE_MLOCK and RELEASE_MLOCK.

If a CPU wants to acquire a lock, then it reads this register. If the value returned is 0, it has successfully acquired the MLOCK. A return value of 1 indicates that the acquire failed. At any time, the CPU can write 0 to MLOCK, releasing the MLOCK (note that hardware does not check if the CPU owned the lock). If the CPU writes a 1 to MLOCK, this is undefined and is ignored by the hardware (a NOP). Use MLOCK_OWNER to read the status of a lock (since reading MLOCK itself has a side effect).

Offset: 0x2c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_0

13.9.73 HOST1X_SYNC_MLOCK_1_0

Offset: 0x2c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_1

13.9.74 HOST1X_SYNC_MLOCK_2_0

Offset: 0x2c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_2

13.9.75 HOST1X_SYNC_MLOCK_3_0

Offset: 0x2cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_3

13.9.76 HOST1X_SYNC_MLOCK_4_0

Offset: 0x2d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_4

13.9.77 HOST1X_SYNC_MLOCK_5_0

Offset: 0x2d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_5

13.9.78 HOST1X_SYNC_MLOCK_6_0

Offset: 0x2d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_6

13.9.79 HOST1X_SYNC_MLOCK_7_0

Offset: 0x2dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_7

13.9.80 HOST1X_SYNC_MLOCK_8_0

Offset: 0x2e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_8

13.9.81 HOST1X_SYNC_MLOCK_9_0

Offset: 0x2e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_9

13.9.82 HOST1X_SYNC_MLOCK_10_0

Offset: 0x2e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_10

13.9.83 HOST1X_SYNC_MLOCK_11_0

Offset: 0x2ec | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_11

13.9.84 HOST1X_SYNC_MLOCK_12_0

Offset: 0x2f0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_12

13.9.85 HOST1X_SYNC_MLOCK_13_0

Offset: 0x2f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_13

13.9.86 HOST1X_SYNC_MLOCK_14_0

Offset: 0x2f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_14

13.9.87 HOST1X_SYNC_MLOCK_15_0

Offset: 0x2fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	MLOCK_15

13.9.88 HOST1X_SYNC_MLOCK_OWNER_0_0

MLOCK_OWNER is a read-only status for MLOCK. When MLOCK_*_OWNS are all zeros, it indicates that the MLOCK is free (zero). When MLOCK has been acquired (set), then one of MLOCK_*_OWNS will be non-zero.

Either bit 0 or 1 will be set -- indicating whether a channel or a CPU has acquired the MLOCK. If a channel owns the MLOCK, then the channel number is given by the MLOCK_OWNER_CHID field.

Offset: 0x340 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_0
1	X	MLOCK_CPU_OWNS_0
0	X	MLOCK_CH_OWNS_0

13.9.89 HOST1X_SYNC_MLOCK_OWNER_1_0

Offset: 0x344 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_1
1	X	MLOCK_CPU_OWNS_1
0	X	MLOCK_CH_OWNS_1

13.9.90 HOST1X_SYNC_MLOCK_OWNER_2_0

Offset: 0x348 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_2
1	X	MLOCK_CPU_OWNS_2
0	X	MLOCK_CH_OWNS_2

13.9.91 HOST1X_SYNC_MLOCK_OWNER_3_0

Offset: 0x34c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_3
1	X	MLOCK_CPU_OWNS_3
0	X	MLOCK_CH_OWNS_3

13.9.92 HOST1X_SYNC_MLOCK_OWNER_4_0

Offset: 0x350 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_4
1	X	MLOCK_CPU_OWNS_4
0	X	MLOCK_CH_OWNS_4

13.9.93 HOST1X_SYNC_MLOCK_OWNER_5_0

Offset: 0x354 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_5
1	X	MLOCK_CPU_OWNS_5
0	X	MLOCK_CH_OWNS_5

13.9.94 HOST1X_SYNC_MLOCK_OWNER_6_0

Offset: 0x358 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_6
1	X	MLOCK_CPU_OWNS_6
0	X	MLOCK_CH_OWNS_6

13.9.95 HOST1X_SYNC_MLOCK_OWNER_7_0

Offset: 0x35c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_7
1	X	MLOCK_CPU_OWNS_7
0	X	MLOCK_CH_OWNS_7

13.9.96 HOST1X_SYNC_MLOCK_OWNER_8_0

Offset: 0x360 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_8
1	X	MLOCK_CPU_OWNS_8
0	X	MLOCK_CH_OWNS_8

13.9.97 HOST1X_SYNC_MLOCK_OWNER_9_0

Offset: 0x364 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_9
1	X	MLOCK_CPU_OWNS_9
0	X	MLOCK_CH_OWNS_9

13.9.98 HOST1X_SYNC_MLOCK_OWNER_10_0

Offset: 0x368 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_10
1	X	MLOCK_CPU_OWNS_10
0	X	MLOCK_CH_OWNS_10

13.9.99 HOST1X_SYNC_MLOCK_OWNER_11_0

Offset: 0x36c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_11
1	X	MLOCK_CPU_OWNS_11
0	X	MLOCK_CH_OWNS_11

13.9.100 HOST1X_SYNC_MLOCK_OWNER_12_0

Offset: 0x370 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_12
1	X	MLOCK_CPU_OWNS_12
0	X	MLOCK_CH_OWNS_12

13.9.101 HOST1X_SYNC_MLOCK_OWNER_13_0

Offset: 0x374 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_13
1	X	MLOCK_CPU_OWNS_13
0	X	MLOCK_CH_OWNS_13

13.9.102 HOST1X_SYNC_MLOCK_OWNER_14_0

Offset: 0x378 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_14
1	X	MLOCK_CPU_OWNS_14
0	X	MLOCK_CH_OWNS_14

13.9.103 HOST1X_SYNC_MLOCK_OWNER_15_0

Offset: 0x37c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	MLOCK_OWNER_CHID_15
1	X	MLOCK_CPU_OWNS_15
0	X	MLOCK_CH_OWNS_15

13.9.104 HOST1X_SYNC_MLOCK_ERROR_0_0

If a channel attempts to release an MLOCK that it does not own, then the release has no effect on MLOCK, but the corresponding error bit is set (this includes releasing an MLOCK owned by no one).

Offset: 0x3c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	0x0	MLOCK_ERROR_15
14	0x0	MLOCK_ERROR_14
13	0x0	MLOCK_ERROR_13
12	0x0	MLOCK_ERROR_12
11	0x0	MLOCK_ERROR_11
10	0x0	MLOCK_ERROR_10
9	0x0	MLOCK_ERROR_9
8	0x0	MLOCK_ERROR_8
7	0x0	MLOCK_ERROR_7
6	0x0	MLOCK_ERROR_6
5	0x0	MLOCK_ERROR_5
4	0x0	MLOCK_ERROR_4
3	0x0	MLOCK_ERROR_3
2	0x0	MLOCK_ERROR_2
1	0x0	MLOCK_ERROR_1
0	0x0	MLOCK_ERROR_0

13.9.105 HOST1X_SYNC_SYNCPT_0_0

Note: SYNCPT REGISTERS
Although the CPU can read and write these registers, typically they are modified and compared using increment syncpt client methods and wait syncpt host methods

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

Bit	Reset	Description
31:0	X	SYNCPT_0

13.9.106 HOST1X_SYNC_SYNCPT_1_0

Offset: 0x404 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_1

13.9.107 HOST1X_SYNC_SYNCPT_2_0

Offset: 0x408 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_2

13.9.108 HOST1X_SYNC_SYNCPT_3_0

Offset: 0x40c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_3

13.9.109 HOST1X_SYNC_SYNCPT_4_0

Offset: 0x410 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_4

13.9.110 HOST1X_SYNC_SYNCPT_5_0

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

Bit	Reset	Description
31:0	X	SYNCPT_5

13.9.111 HOST1X_SYNC_SYNCPT_6_0

Offset: 0x418 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_6

13.9.112 HOST1X_SYNC_SYNCPT_7_0

Offset: 0x41c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_7

13.9.113 HOST1X_SYNC_SYNCPT_8_0

Offset: 0x420 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_8

13.9.114 HOST1X_SYNC_SYNCPT_9_0

Offset: 0x424 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_9

13.9.115 HOST1X_SYNC_SYNCPT_10_0

Offset: 0x428 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_10

13.9.116 HOST1X_SYNC_SYNCPT_11_0

Offset: 0x42c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_11

13.9.117 HOST1X_SYNC_SYNCPT_12_0

Offset: 0x430 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_12

13.9.118 HOST1X_SYNC_SYNCPT_13_0

Offset: 0x434 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_13

13.9.119 HOST1X_SYNC_SYNCPT_14_0

Offset: 0x438 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_14

13.9.120 HOST1X_SYNC_SYNCPT_15_0

Offset: 0x43c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_15

13.9.121 HOST1X_SYNC_SYNCPT_16_0

Offset: 0x440 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_16

13.9.122 HOST1X_SYNC_SYNCPT_17_0

Offset: 0x444 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_17

13.9.123 HOST1X_SYNC_SYNCPT_18_0

Offset: 0x448 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_18

13.9.124 HOST1X_SYNC_SYNCPT_19_0

Offset: 0x44c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_19

13.9.125 HOST1X_SYNC_SYNCPT_20_0

Offset: 0x450 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_20

13.9.126 HOST1X_SYNC_SYNCPT_21_0

Offset: 0x454 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_21

13.9.127 HOST1X_SYNC_SYNCPT_22_0

Offset: 0x458 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_22

13.9.128 HOST1X_SYNC_SYNCPT_23_0

Offset: 0x45c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_23

13.9.129 HOST1X_SYNC_SYNCPT_24_0

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

Bit	Reset	Description
31:0	X	SYNCPT_24

13.9.130 HOST1X_SYNC_SYNCPT_25_0

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

Bit	Reset	Description
31:0	X	SYNCPT_25

13.9.131 HOST1X_SYNC_SYNCPT_26_0

Offset: 0x468 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_26

13.9.132 HOST1X_SYNC_SYNCPT_27_0

Offset: 0x46c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_27

13.9.133 HOST1X_SYNC_SYNCPT_28_0

Offset: 0x470 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_28

13.9.134 HOST1X_SYNC_SYNCPT_29_0

Offset: 0x474 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_29

13.9.135 HOST1X_SYNC_SYNCPT_30_0

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

Bit	Reset	Description
31:0	X	SYNCPT_30

13.9.136 HOST1X_SYNC_SYNCPT_31_0

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

Bit	Reset	Description
31:0	X	SYNCPT_31

13.9.137 HOST1X_SYNC_SYNCPT_INT_THRESH_0_0

Syncpt interrupt thresholds

When a syncpt threshold interrupt is enabled, an interrupt is signaled when (SYNCPT_THRESH_INT_MASK[indx] == cpu) && (SYNCPT[indx] >= SYNCPT_INT_THRESH[indx]), where the comparison takes into account wrapping, and the interrupt is routed to the specified CPU.

Offset: 0x500 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_0

13.9.138 HOST1X_SYNC_SYNCPT_INT_THRESH_1_0

Offset: 0x504 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_1

13.9.139 HOST1X_SYNC_SYNCPT_INT_THRESH_2_0

Offset: 0x508 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_2

13.9.140 HOST1X_SYNC_SYNCPT_INT_THRESH_3_0

Offset: 0x50c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_3

13.9.141 HOST1X_SYNC_SYNCPT_INT_THRESH_4_0

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

Bit	Reset	Description
31:0	X	INT_THRESH_4

13.9.142 HOST1X_SYNC_SYNCPT_INT_THRESH_5_0

Offset: 0x514 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_5

13.9.143 HOST1X_SYNC_SYNCPT_INT_THRESH_6_0

Offset: 0x518 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_6

13.9.144 HOST1X_SYNC_SYNCPT_INT_THRESH_7_0

Offset: 0x51c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_7

13.9.145 HOST1X_SYNC_SYNCPT_INT_THRESH_8_0

Offset: 0x51c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_8

13.9.146 HOST1X_SYNC_SYNCPT_INT_THRESH_9_0

Offset: 0x524 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_9

13.9.147 HOST1X_SYNC_SYNCPT_INT_THRESH_10_0

Offset: 0x528 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_10

13.9.148 HOST1X_SYNC_SYNCPT_INT_THRESH_11_0

Offset: 0x52c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_11

13.9.149 HOST1X_SYNC_SYNCPT_INT_THRESH_12_0

Offset: 0x530 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_12

13.9.150 HOST1X_SYNC_SYNCPT_INT_THRESH_13_0

Offset: 0x534 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_13

13.9.151 HOST1X_SYNC_SYNCPT_INT_THRESH_14_0

Offset: 0x538 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_14

13.9.152 HOST1X_SYNC_SYNCPT_INT_THRESH_15_0

Offset: 0x53c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_15

13.9.153 HOST1X_SYNC_SYNCPT_INT_THRESH_16_0

Offset: 0x540 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_16

13.9.154 HOST1X_SYNC_SYNCPT_INT_THRESH_17_0

Offset: 0x544 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_17

13.9.155 HOST1X_SYNC_SYNCPT_INT_THRESH_18_0

Offset: 0x548 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_18

13.9.156 HOST1X_SYNC_SYNCPT_INT_THRESH_19_0

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

Bit	Reset	Description
31:0	X	INT_THRESH_19

13.9.157 HOST1X_SYNC_SYNCPT_INT_THRESH_20_0

Offset: 0x550 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_20

13.9.158 HOST1X_SYNC_SYNCPT_INT_THRESH_21_0

Offset: 0x554 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_21

13.9.159 HOST1X_SYNC_SYNCPT_INT_THRESH_22_0

Offset: 0x558 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_22

13.9.160 HOST1X_SYNC_SYNCPT_INT_THRESH_23_0

Offset: 0x55c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_23

13.9.161 HOST1X_SYNC_SYNCPT_INT_THRESH_24_0

Offset: 0x560 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_24

13.9.162 HOST1X_SYNC_SYNCPT_INT_THRESH_25_0

Offset: 0x564 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_25

13.9.163 HOST1X_SYNC_SYNCPT_INT_THRESH_26_0

Offset: 0x568 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_26

13.9.164 HOST1X_SYNC_SYNCPT_INT_THRESH_27_0

Offset: 0x56c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_27

13.9.165 HOST1X_SYNC_SYNCPT_INT_THRESH_28_0

Offset: 0x570 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_28

13.9.166 HOST1X_SYNC_SYNCPT_INT_THRESH_29_0

Offset: 0x574 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_29

13.9.167 HOST1X_SYNC_SYNCPT_INT_THRESH_30_0

Offset: 0x578 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_30

13.9.168 HOST1X_SYNC_SYNCPT_INT_THRESH_31_0

Offset: 0x57c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INT_THRESH_31

13.9.169 HOST1X_SYNC_SYNCPT_BASE_0_0

Syncpt base registers are used by wait_syncpt_base method. The wait will be released when SYNCPT[indx] >= (BASE[base_indx] + offset), where indx, base_indx, and offset are supplied by the wait method.

There are 12 Syncpt Base registers, where n = 0 through 11.

Offset: 0x600 + (n * 0x4) │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BASE_n

13.9.170 HOST1X_SYNC_SYNCPT_CPU_INCR_0

When a CPU writes the vector to this register, if vector[i] is set, then syncpt[i] is incremented. Read returns undefined value.

Offset: 0x700 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SYNCPT_CPU_INCR_VECTOR

13.10 Host Class Methods

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

Class offsets are always relative to the class (based at 0).

13.10.1 NV_CLASS_HOST_INCR_SYNCPT_0

All Classes have the INCR_SYNCPT method. For host, this method, immediately increments SYNCPT[indx], irrespective of the condition. Note that INCR_SYNCPT_CNTRL and INCR_SYNCPT_ERROR are included for consistency with host clients, but writes to INCR_SYNCPT_CNTRL have no effect on the operation of Host1x, and because there are no condition FIFOs to overflow, INCR_SYNCPT_ERROR will never be set.

Offset: 0x0 │ Byte Offset: 0x0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12

Bit	Reset	Description
		13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	INDX: syncpt index value

13.10.2 NV_CLASS_HOST_INCR_SYNCPT_CNTRL_0

Offset: 0x1 │ Byte Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

13.10.3 NV_CLASS_HOST_INCR_SYNCPT_ERROR_0

Offset: 0x2 │ Byte Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

13.10.4 NV_CLASS_HOST_WAIT_SYNCPT_0

Wait on syncpt method.

Command dispatch will stall until:

SYNCPT[indx][NV_HOST1X_SYNCPT_THRESH_WIDTH-1:0] >= threshold[NV_HOST1X_SYNCPT_THRESH_WIDTH-1:0]

The comparison takes into account the possibility of wrapping. Note that more bits are allocated for index and threshold than may be used in an implementation. Use NV_HOST1X_SYNCPT_NB_PTS for the number of syncpts, and NV_HOST1X_SYNCPT_THRESH_WIDTH for the number of bits used by the comparison.

Offset: 0x8 │ Byte Offset: 0x20 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	INDX
23:0	X	THRESH

13.10.5 NV_CLASS_HOST_WAIT_SYNCPT_BASE_0

Wait on syncpt method using base register.

Command dispatch will stall until:

SYNCPT[indx][NV_HOST1X_SYNCPT_THRESH_WIDTH-1:0] >= (SYNCPT_BASE[base_indx]+offset)

The comparison takes into account the possibility of wrapping. Note that more bits are allocated for INDX and BASE_INDX than may be used in an implementation. Use NV_HOST1X_SYNCPT_NB_PTS for the number of syncpts, Use NV_HOST1X_SYNCPT_NB_BASES for the number of syncpt_bases, and NV_HOST1X_SYNCPT_THRESH_WIDTH for the

number of bits used by the comparison If NV_HOST1X_SYNCPT_THRESH_WIDTH is greater than 16, the offset is sign-extended before it is added to SYNCPT_BASE.

Offset: 0x9 | Byte Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	INDX
23:16	X	BASE_INDX
15:0	X	OFFSET

13.10.6 NV_CLASS_HOST_WAIT_SYNCPT_INCR_0

Wait on syncpt increment method.

Command dispatch will stall until the next time that SYNCPT[indx] is incremented.

Note that more bits are allocated for INDX than may be used in an implementation. Use NV_HOST1X_SYNCPT_NB_PTS for the number of syncpts.

Offset: 0xa | Byte Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	INDX

13.10.7 NV_CLASS_HOST_LOAD_SYNCPT_BASE_0

Load syncpt base method.

SYNCPT_BASE[indx] = value

Offset: 0xb | Byte Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	BASE_INDX
23:0	X	VALUE

13.10.8 NV_CLASS_HOST_INCR_SYNCPT_BASE_0

Increment syncpt base method.

SYNCPT_BASE[indx] += offset

Offset: 0xc | Byte Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	BASE_INDX
23:0	X	OFFSET

13.10.9 NV_CLASS_HOST_CLEAR_0

Clear method. Any bits set in VECTOR will be cleared in the channel's RAISE vector.

Offset: 0xd | Byte Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VECTOR

13.10.10 NV_CLASS_HOST_WAIT_0

Wait method. Command dispatch will stall until any of the bits set in VECTOR become set in the channel's RAISE vector.

Offset: 0xe │ Byte Offset: 0x38 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VECTOR

13.10.11 NV_CLASS_HOST_WAIT_WITH_INTR_0

Wait with Interrupt method. Identical to the WAIT method except an interrupt will be triggered when the WAIT requirement is satisfied. This is obsolete and preserved here only for completeness.

Offset: 0xf │ Byte Offset: 0x3c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VECTOR

13.10.12 NV_CLASS_HOST_DELAY_USEC_0

Delay number of microseconds. Command dispatch will stall until the number of microseconds indicated in NUSEC has passed. The timing of microseconds is controlled by the USEC_CLK register.

Offset: 0x10 │ Byte Offset: 0x40 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:0	X	NUSEC: Enough for 1.05 seconds

13.10.13 NV_CLASS_HOST_TICKCOUNT_HI_0

This register value will initialize the high 32 bits of the tick count value in the host clock counter.

Offset: 0x11 │ Byte Offset: 0x44 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_HI: Read or write tick count

13.10.14 NV_CLASS_HOST_TICKCOUNT_LO_0

This register value will initialize the low 32 bits of the tick count value in the host clock counter.

Offset: 0x12 │ Byte Offset: 0x48 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_LO: Read or write tick count

13.10.15 NV_CLASS_HOST_TICKCTRL_0

This register write enables the tick counter on the host clock to start counting.

Offset: 0x13 │ Byte Offset: 0x4c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TICKCNT_ENABLE: Enable or Disable tick counter 0 = DISABLE 1 = ENABLE

13.10.16 NV_CLASS_HOST_INDCTRL_0

Indirect Addressing

These registers (along with INDDATA) are used to indirectly read/write either register or memory. Host registers are not accessible using this interface. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA.

Either INDCTRL/INDOFF2 or INDOFF can be used, but INDOFF may not be able to address all memory in chips with large memory maps. The redundant bits in INDCTRL and INDOFF are shared, so writing either offset sets those bits.

Note: The following restrictions apply to the use of indirect memory writes:

- At initialization time, do a dummy indirect write (with all byte enables set to zero)
- Dedicate an MLOCK for indirect memory writes, then before a channel issues a set of indirect memory writes it must acquire this MLOCK; after the writes have been issued, the MLOCK is released -- this will restrict the use of indirect memory writes to a single channel at a time.

Offset: 0x2b │ Byte Offset: 0xac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	INDBE: Byte enables. Will apply to all subsequent data transactions. Not applicable for reads.
27	X	AUTOINC: Auto increment of read/write address 0 = DISABLE 1 = ENABLE
26	X	SPOOL: Route return data to spool FIFO, only applicable to reads 0 = DISABLE 1 = ENABLE
1	X	ACCTYPE: Access type: indirect register or indirect frame buffer 0 = REG 1 = FB
0	X	RWN: Read/write 0 = WRITE 1 = READ

13.10.17 NV_CLASS_HOST_INDOFF2_0

Offset: 0x2c │ Byte Offset: 0xb0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:18	X	INDMODID: ACCTYPE=REG: Register module ID 0 = HOST1X 1 = MPE 2 = VI 3 = EPP 4 = ISP 5 = GR2D 6 = GR3D 8 = DISPLAY 9 = DISPLAYB 10 = HDMI 12 = DSI 16 = DSIB 19 = MSENC 20 = TSEC
31:2	X	INDOFFSET: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET: ACCTYPE=REG: register offset ([15:0])

13.10.18 NV_CLASS_HOST_INDOFF_0

Offset: 0x2d | Byte Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	INDBE: Byte enables. Will apply to all subsequent data transactions. Not applicable for reads.
27	X	AUTOINC: Auto increment of read/write address 0 = DISABLE 1 = ENABLE
26	X	SPOOL: Route return data to spool FIFO, only applicable to reads 0 = DISABLE 1 = ENABLE
25:18	X	INDMODID: ACCTYPE=REG: register module ID 0 = HOST1X 1 = MPE 2 = VI 3 = EPP 4 = ISP 5 = GR2D 6 = GR3D 8 = DISPLAY 9 = DISPLAYB 12 = DSI 10 = HDMI 16 = DSIB 19 = MSENC 20 = TSEC
25:2	X	INDOFFSET: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET: ACCTYPE=REG: register offset ([15:0])
1	X	ACCTYPE: Access type: indirect register or indirect frame buffer 0 = REG 1 = FB
0	X	RWN: Read/write 0 = WRITE 1 = READ

13.10.19 NV_CLASS_HOST_INDDATA_0

These registers, when written, either write to the data to the INDOFFSET in INDOFF or trigger a read of the offset at INDOFFSET. This is an array of 31 identical register entries; the register fields below apply to each entry.

Offset: 0x2e..0x4c | Byte Offset: 0xb8..0x130 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INDDATA: Read or write data

13.10.20 NV_CLASS_HOST_LOAD_SYNCPT_PAYLOAD_32_0

Offset: 0x4e | Byte Offset: 0x138 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CHANNEL_SYNCPT_PAYLOAD

13.10.21 NV_CLASS_HOST_STALLCTRL_0

Offset: 0x4f | Byte Offset: 0x13c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0X0	ENABLE

13.10.22 NV_CLASS_HOST_WAIT_SYNCPT_32_0

Offset: 0x50 | Byte Offset: 0x140 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	INDX

13.10.23 NV_CLASS_HOST_WAIT_SYNCPT_BASE_32_0

Offset: 0x51 | Byte Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	BASE_INDX
7:0	X	INDX

13.10.24 NV_CLASS_HOST_LOAD_SYNCPT_BASE_32_0

Offset: 0x52 | Byte Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	INDX

13.10.25 NV_CLASS_HOST_INCR_SYNCPT_BASE_32_0

Offset: 0x53 | Byte Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	INDX

13.10.26 NV_CLASS_HOST_STALLCOUNT_HI_0

Offset: 0x54 | Byte Offset: 0x150 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0X0	STALLCOUNT_HI

13.10.27 NV_CLASS_HOST_STALLCOUNT_LO_0

Offset: 0x55 | Byte Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0X0	STALLCOUNT_LO

13.10.28 NV_CLASS_HOST_XFERCTRL_0

Offset: 0x56 | Byte Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0X0	ENABLE

13.10.29 NV_CLASS_HOST_CHANNEL_XFER_HI_0

Offset: 0x57 | Byte Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0X0	CHANNEL_XFER_HI

13.10.30 NV_CLASS_HOST_CHANNEL_XFER_LO_0

Offset: 0x58 | Byte Offset: 0x160 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0X0	CHANNEL_XFER_LO

13.11 Host Proto Channel Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

13.11.1 HOST1X_PROTCHANNEL_FIFOSTAT_0

CFNUMEMPTY is the number of free slots available in the per-channel command FIFO (needed for PIO or polling for completion of a wait).

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

Bit	Reset	Description
31	X	INDRDY: Indicates that INDCOUNT==0, so it should be OK to issue another read
28:24	X	OUTFENTRIES: Number of entries available for reading in this channel's output FIFO
20:16	X	REGFNUMEMPTY: Register write/read FIFO free count
13	X	CFGATHER3D: Indicates whether GATHER3D is active. If a GATHER3D command issued via PIO, software must wait for the GATHER3D to be IDLE before issuing another command. 0 = IDLE 1 = BUSY
12	X	CFGATHER: Indicates whether or not GATHER is active. If a GATHER command issued via PIO, software must wait for the GATHER to be IDLE before issuing another command. 0 = IDLE 1 = BUSY
11	X	CFEMPTY: Indicates whether the command FIFO is empty or not 0 = NOTEMPTY 1 = EMPTY
10:0	X	CFNUMEMPTY: Command FIFO free count

13.11.2 HOST1X_PROTCHANNEL_INDOFF_0

The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA.

The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set the other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases. For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.

Indirect frame buffer writes are STRONGLY DISCOURAGED. There are better ways to write to memory (direct and through the channel memory map) and there is limited flow control in the host. It is very easy to get into trouble with indirect frame buffer writes.

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTOINC: Auto increment of read/write address 0 = DISABLE 1 = ENABLE
30	X	ACCTYPE: access type: indirect register or indirect frame buffer 0 = REG 1 = FB
29	X	BUF32B: Buffer up 32 bits of register data before sending it. Otherwise, register writes will be sent as soon as they are received. Does not support byte writes in 16-bit host. Does not affect frame buffer writes. 0 = NOBUF 1 = BUF
28:27	X	INDSWAP: Indirect frame buffer access swap control. 00 = No byte swap 01 = 16-bit byte swap ([31:0] -> {[23:16],[31:24],[7:0],[15:8]}) 10 = 32-bit byte swap ([31:0] -> {[7:0],[15:8],[23:16],[31:24]}) 11 = 32-bit word swap ([31:0] -> {[15:8],[7:0],[31:24],[23:16]}) 0 = NONE 1 = BYTE16 2 = BYTE32 3 = WORD32
25:18	X	INDMODID: ACCTYPE=REG: register module ID 0 = HOST1X 1 = MPE 2 = VI 3 = EPP 4 = ISP 5 = GR2D 6 = GR3D 8 = DISPLAY 9 = DISPLAYB 12 = DSI 10 = HDMI 16 = DSIB 19 = MSENC 20 = TSEC
25:2	X	INDOFFSET: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET: ACCTYPE=REG: register offset ([15:0])
0	X	INDOFFUPD: Optionally disable the update of INDOFFSET when writing this register 0 = UPDATE 1 = NO_UPDATE

13.11.3 HOST1X_PROTCHANNEL_INDCNT_0

Indirect register access count

Used to trigger indirect reads. Holds the number of registers/memory locations that will be read out. Channels should not request more than there is space available in their output FIFO. Only the protected channel should make liberal use of this feature for speeding up context switching.

For indirect frame buffer reads, each channel cannot issue more than NV_HOST1X_MAX_IND_FB_READS at once. The read data must return and be written into the per-channel output FIFO before any additional reads can be issued.

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0X0	INDCOUNT

13.11.4 HOST1X_PROTCHANNEL_INDDATA_0

This register, when written, writes the data to the INDOFFSET in INDOFF. For reads, a REGFNUMEMPTY number of 32-bit values can be read before needing to poll FIFOSTAT again.

The per-channel output FIFO (OUTFENTRIES) is readable via this offset. A read of INDDATA will pop an entry off of the per-channel output FIFO.

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	0X0	INDDATA: Read or write data

13.11.5 HOST1X_PROTCHANNEL_INDOFF2_0

Note: This spec file contains additions to the "common" registers that cannot be put in the common section, otherwise all of the other registers will shift.

The INDOFF and INDOFF2 registers (along with INDCNT and INDDATA) are used to indirectly read/write modules outside the host. If AUTOINC is set, INDOFFSET increments by 4 on every access of INDDATA. REGFNUMEMPTY is polled to determine when valid data can be read from INDDATA.

The INDOFF register has limited capability on chips with large memory maps. If the top bit of the memory address is >= 27, all of memory cannot be addressed with INDOFF. In these cases, use INDOFF2 to set the offset while still using INDOFF to set the other parameters. Always have INDOFFUPD set to NO_UPDATE in these cases. For register accesses, using INDOFF (with INDOFFUPD set to UPDATE) is always more efficient, since it only requires one write.

Indirect frame buffer writes are STRONGLY DISCOURAGED. There are better ways to write to memory (direct and through the channel memory map) and there is limited flow control in the host. It is very easy to get into trouble with indirect frame buffer writes.

Offset: 0x8c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:18	X	INDMODID2: ACCTYPE=REG: register module ID 0 = HOST1X 1 = MPE 2 = VI 3 = EPP 4 = ISP 5 = GR2D 6 = GR3D

Bit	Reset	Description
		8 = DISPLAY 9 = DISPLAYB 12 = DSI 10 = HDMI 16 = DSIB 19 = MSENC 20 = TSEC
31:2	X	INDOFFSET2: ACCTYPE=FB: frame buffer address
17:2	X	INDROFFSET2: ACCTYPE=REG: register offset ([15:0])

13.11.6 HOST1X_PROTCHANNEL_TICKCOUNT_HI_0

This register holds the high 32 bits of the tick count value

Offset: 0x90 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_HI

13.11.7 HOST1X_PROTCHANNEL_TICKCOUNT_LO_0

This register holds the low 32 bits of the tick count value

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_LO

13.11.8 HOST1X_PROTCHANNEL_CHANNELCTRL_0

This register will be used for controlling some channel-related commands including enabling/disabling of the tick counter.

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	ENABLETICKCNT: Enable or disable tick counter 0 = DISABLE 1 = ENABLE

13.11.9 HOST1X_PROTCHANNEL_PAYLOAD_0

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	PAYLOAD

13.11.10 HOST1X_PROTCHANNEL_STALLCTRL_0

Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE_CHANNEL_STALL

13.11.11 HOST1X_PROTCHANNEL_STALLCOUNT_HI_0

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	STALLCOUNT_HI

13.11.12 HOST1X_PROTCHANNEL_STALLCOUNT_LO_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	STALLCOUNT_LO

13.11.13 HOST1X_PROTCHANNEL_XFERCTRL_0

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	ENABLE_CHANNEL_XFER

13.11.14 HOST1X_PROTCHANNEL_CHANNEL_XFER_HI_0

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CHANNEL_XFER_HI

13.11.15 HOST1X_PROTCHANNEL_CHANNEL_XFER_LO_0

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CHANNEL_XFER_LO

13.11.16 HOST1X_PROTCHANNEL_HOST1X_CHANNEL_SPARE_0

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00ff0000 (0b00000000111111110000000000000000)

Bit	Reset	Description
31:16	0xff	CHANNEL_SPARE_HI
15:0	0x0	CHANNEL_SPARE_LO

13.11.17 HOST1X_PROTCHANNEL_TICKCOUNT_THRESHOLD_HI_0

This register holds the high 32 bits of the tick count threshold value

Offset: 0xbc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_THRESHOLD_HI

13.11.18 HOST1X_PROTCHANNEL_TICKCOUNT_THRESHOLD_LO_0

This register holds the low 32 bits of the tick count threshold value

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TICKS_THRESHOLD_LO

13.11.19 HOST1X_PROTCHANNEL_TICKCOUNT_THRESHOLD_CTRL_0

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	THRES_CMP: Enable or disable tick threshold compare 0 = DISABLE 1 = ENABLE

14.0 GR2D

14.1 Introduction

GR2D is a 2D graphics acceleration block, capable of performing a range of copying, blending, scaling, rotation, and other 2D graphics operations. It is also sometimes used in conjunction with the Encode Pre-Processor (EPP), to which it has a direct path.

This section is not intended to be a programming guide to GR2D, as it is expected that NVIDIA supplied drivers will always be used with it. However, the register interface is documented to aid with understanding those drivers.

14.2 Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

14.2.1 G2SB_G2SBFORMAT_0

Offset: 0x1c │ Byte Offset: 0x70 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12:8	X	DIFMT: Destination Image Data Format. This parameter defines the data format of destination output. There are two groups of data formats, RGB and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement. 00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 01000: bpp16 5-6-5 {R[4:0], G[5:0], B[4:0]} 01001: RESERVED 01010: RESERVED 01011: RESERVED 01100: bpp16 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]} 01101: RESERVED 01110: R8G8B8A8 01111: B8G8R8A8 1xxxx: RESERVED 0 = U8Y8V8Y8_OB 1 = Y8U8Y8V8_OB 2 = Y8V8Y8U8_OB 3 = V8Y8U8Y8_OB 4 = U8Y8V8Y8_TC 5 = Y8U8Y8V8_TC 6 = Y8V8Y8U8_TC 7 = V8Y8U8Y8_TC 8 = B5G6R5 9 = RESERVED9 10 = RESERVED10 11 = RESERVED11 12 = B5G6R5BS 13 = RESERVED13 14 = R8G8B8A8 15 = B8G8R8A8 16 = RESERVED16

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>17 = RESERVED17
18 = RESERVED18
19 = RESERVED19
20 = RESERVED20
21 = RESERVED21
22 = RESERVED22
23 = RESERVED23
24 = RESERVED24
25 = RESERVED25
26 = RESERVED26
27 = RESERVED27
28 = RESERVED28
29 = RESERVED29
30 = RESERVED30
31 = RESERVED31</td></tr>
<tr><td>4:0</td><td>X</td><td>SIFMT: This parameter defines the data format of source input.
There are two groups of data formats, RGB format and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement.
00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
01000: B5G6R5 5-6-5 {R[4:0], G[5:0], B[4:0]}
01001: RESERVED
01010: RESERVED
01011: RESERVED
01100: B5G6R5 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]}
01101: RESERVED
01110: R8G8B8A8
01111: B8G8R8A8
1xxxx: RESERVED
StretchBlit Inputs: RGB inputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV420 input = {YUV420 is converted into 4:2:2 UYVY via memory client} YUV422 inputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}
StretchBlit Outputs: RGB outputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV422 outputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}
StretchBlit Input/Output Rules
<table>
<tr><th>src format</th><th>internal sb format</th><th>dst format</th></tr>
<tr><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td><td>R8G8B8A8</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
</table>
StretchBlit color space converter supports YUV->RGB, YUV->YUV (gain) and RGB->RGB (gain). There is no support for RGB->YUV.
0 = U8Y8V8Y8_OB
1 = Y8U8Y8V8_OB
2 = Y8V8Y8U8_OB
3 = V8Y8U8Y8_OB
4 = U8Y8V8Y8_TC
5 = Y8U8Y8V8_TC</td></tr>
</table>

Bit	Reset	Description
		6 = Y8V8Y8U8_TC 7 = V8Y8U8Y8_TC 8 = B5G6R5 9 = RESERVED9 10 = RESERVED10 11 = RESERVED11 12 = B5G6R5BS 13 = RESERVED13 14 = R8G8B8A8 15 = B8G8R8A8 16 = RESERVED16 17 = RESERVED17 18 = RESERVED18 19 = RESERVED19 20 = RESERVED20 21 = RESERVED21 22 = RESERVED22 23 = RESERVED23 24 = RESERVED24 25 = RESERVED25 26 = RESERVED26 27 = RESERVED27 28 = RESERVED28 29 = RESERVED29 30 = RESERVED30 31 = RESERVED31

14.2.2 G2SB_G2CONTROLSB_0

Offset: 0x1d │ Byte Offset: 0x74 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000xxx)

Bit	Reset	Description
31	0x0	DISDW: Output destination writes go either to image memory or the EPP. 0: Output data is sent to memory 1: YUV or RGB data is passed directly to the EPP module, and no destination writes take place. 0 = GOTO_IMAGE_BUFFER 1 = GOTO_EPP
30	0x0	ENDITH: Enable Dithering (ENDITH) For 16-bit RGB output modes, the LSB of the color components can be modified by adding a variable residual value that will reduce the banding artifacts that can appear on the display. 0: Normal operation 1: Enable dithering 0 = DISABLE 1 = ENABLE
28	0x0	KPOL: Key Signal Polarity (KPOL) Color/Chroma key signal is generated by comparing source input pixel color to a range of a color specified by lower and upper limit values. The key signal is interpreted in two ways, depending on which one of video and graphics images is foreground (and the other is background). This is effective only if Key signal generator is enabled. (see G2CMKEYL, G2CMKEYU) 0: Key signal is set to 1 when source pixel is within the lower and upper limit color range. 1: Key signal is set to 1 when source pixel is outside the lower and upper limit color range. 0 = WITHIN_BOUNDS 1 = OUTSIDE_BOUNDS
27	0x0	KEYEN: Key Signal Generator Enable (KEYEN) Key signal generator generates either chroma key signal (from YCbCr signal) or color

Bit	Reset	Description
		key signal (from RGB signal) 0: Key signal generator is disabled. 1: Key signal generator is enabled. 0 = DISABLE 1 = ENABLE
26	0x0	DBSEL: StretchBLT Destination Buffer Selection (DBSEL) StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the destination video area. This bit selects one of the two buffering blocks to which this StretchBLT command delivers the destination image. The two buffering memory blocks are called A and B. 0: Destination image goes to ‘A’ buffer. 1: Destination image goes to ‘B’ buffer. 0 = DST_A 1 = DST_B
25	0x0	SBSEL: StretchBLT Source Buffer Selection (SBSEL) StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the source video area. This bit selects one of the two buffering blocks from which this StretchBLT command receives the source image. The two buffering memory blocks are called source-A and source-B. 0: Source image comes from ‘source-A’ buffer. 1: Source image comes from ‘source-B’ buffer. 0 = SRC_A 1 = SRC_B
24	0x0	SITYPE: StretchBLT Source Type (SITYPE) This bit identifies two types of source images. If source image is 2-to-1 interlaced and StretchBLT processes either one of the two interlaced field-images, physical (positional) displacement between the two interlaced fields must be taken into account. One field-image that is placed higher in position than the other field-image is called top-field and the other is called bottom-field. StretchBLT processing has to lower the top-field (or raise the bottom-field) to match the corresponding two target images in the overlay window (that is progressively scanned) right at the same position. If source image is full frame-image obtained from two interlaced field-images, its type is ‘top-field’. If source images are progressively scanned, the type identification is not significant and they may be designated as one of the two types: 0: Source image is ‘top-field’. 1: Source image is ‘bottom-field’. 0 = TOP_FIELD 1 = BOTTOM_FIELD
23	0x0	RANGEREDFRM: In the VC-1 specification, when the value of RANGEREDFRM variable (1-bit) for a picture is equal to 1, the picture shall be scaled up according to the following equation: Y = clip(((Y-128)*2) + 128);Cb = clip(((Cb-128)*2) + 128);Cr = clip(((Cr-128)*2) + 128);The clip operator limits the output to [0, 255].The input is also limited to [0, 255].In the VC-1 nomenclature, the output (Y, Cb, Cr) tuple corresponds to the 'decoded' picture. The input (Y, Cb, Cr) tuple corresponds to the 'reconstructed' picture. The above equations create the decoded picture while keeping the reconstructed picture intact. Only YUV _OB formats are supported with range reducation enabled, not the YUV _TC or RGB formats. The YUV data coming from the video frame buffers (reconstructed picture) in DRAM was scaled up using the above equations and given to the display. The value of 1-bit wide RANGEREDFRM variable was configured in the DVO per frame and the frame was scaled whenever RANGEREDFRM == 1. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22:20	0x0	HFTYPE: StretchBLT Horizontal Filter Mode (HFTYPE[2:0]) The six-tap horizontal interpolation filter can be operated in various operation modes. For the image expansion, it should be programmed as a pure 6-tap interpolator. For the image contraction, it can work as partly lowpass filter and partly interpolater with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter coefficients for the different modes of horizontal filtering. 000: Pure interpolation filter. 001, 010, 011, 100, 101, 110: mix of interpolation and low pass filters 111: DISABLE. 0 = INTERP 1 = LPF1 2 = LPF2 3 = LPF3 4 = LPF4 5 = LPF5 6 = LPF6 7 = DISABLE
19	0x0	DISCSC: enabled color space converter by default 0 = ENABLE 1 = DISABLE
18	0x0	VFEN: StretchBLT Vertical Filter Enable (VFEN) Vertical filter shall be disabled if source images come from host CPU via CPU Read-FIFO. (SISEL) Vertical filter may be disabled to save some memory bandwidth but this will likely result in degradation of image quality. This option may also be used in the case where source image is progressive scanning and there is no vertical scaling. 0: Vertical filter is disabled. 1: Vertical filter is enabled. 0 = DISABLE 1 = ENABLE
17:16	0x0	VFTYPE: StretchBLT Vertical Filter Mode (VFTYPE[1:0]) This bit is effective when the Vertical Filter is enabled. (bit 18) The two-tap vertical interpolation filter can be operated in various modes. For the image expansion, it should be programmed as a pure 2-tap interpolator. For the image contraction, it can work as partly averager and partly interpolator with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of vertical filtering. 00: Pure interpolation filter. 01: 25% averager, 75% interpolator. 10: 50% averager, 50% interpolator. 11: 100% averager. 0 = INTERP 1 = AVG25_INTERP75 2 = AVG50_INTERP50 3 = AVG
9:8	X	UVST: 00= 1/2 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 2 bytes. 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 4 bytes. 11= use uvstride. **Note**: Use care when using a tiled surface, since 1/2 or 1/4 of luma stride may yield an invalid pitch (stride). 0 = UVS2X 1 = UVS1X 2 = UVS4X 3 = UVS_G2UVSTRIDE

Bit	Reset	Description
7	0x0	ENAHF: enable horizontal alpha filtering if disabled, use the alpha value of third tap for output pixel U,V line stride in 4:2:0 Format. 0 = DISABLE 1 = ENABLE
6	0x0	ENAVF: enable alpha vertical filter 0 = DISABLE 1 = ENABLE
5	0x0	IMODE: Source (input) Data Mode (IMODE). This parameter defines the data mode of source input. 0: Multiplexed Mode (data format is specified by SIFMT[2:0]). One block of source data in the Image Buffer memory contains multiplexed component signals. 1: Planar mode, 4:2:0 YUV (SIFMT[2] specifies data format of U/V components). Three blocks of source data in the Image Buffer memory contain separate Y, U, and V component signals. For a circular buffer input to gr2d, the input format cannot be planar. It must be multiplex, i.e., no YUV420 planar circular buffer input to gr2d. 0 = MULTIPLEX 1 = PLANAR
4	0x0	YUV422PLANAR: Source data is yuv422 planar valid when imode==PLANAR
3	0x0	YUV422ROTATION: YUV422 planar rotated.

14.2.3 G2SB_G2CONTROLSECOND_0

Offset: 0x1e │ Byte Offset: 0x78 │ Read/Write: R/W │ Reset: 0x00000000 (0bxx000000x00xxxx000xxxx000000000x)

Bit	Reset	Description
29	0x0	FR_READWAIT: Fast Rotate wait for reads. Enabling this bit forces FR to wait for the reads to be stored in the data return FIFO before sending writes out to the same block. Always set to ENABLE. This field is debug only. 0 = DISABLE 1 = ENABLE
28:26	0x0	FR_TYPE: Fast Rotate type 000 = FLIP_X 001 = FLIP_Y 010 = TRANS_LR (mirrors about diagonal. Diagonal runs from upper left to lower right) 011 = TRANS_RL (mirrors about diagonal. Diagonal runs from upper right to lower left) 100 = ROT_90 (counterclockwise by 90 degrees) 101 = ROT_180 110 = ROT_270 (clockwise by 90 degrees) 111 = IDENTITY 0 = FLIP_X 1 = FLIP_Y 2 = TRANS_LR 3 = TRANS_RL 4 = ROT_90 5 = ROT_180 6 = ROT_270 7 = IDENTITY
25:24	0x0	FR_MODE: Fast Rotate mode select. G2 Fast Rotate Transforms a surface via FR_TYPE transformation. Works in either 2-buffer (copy) or 1-buffer mode (in place) mode. The engine breaks down a larger surface into a grid of smaller FR_BLOCKs. Works on the granularity of an FR_BLOCK. An FR_BLOCK is: ▪ 16x16 pixel block (DSTCD = bpp8)

Bit	Reset	Description
		▪ 8x8 pixel block (DSTCD = bpp16) ▪ 4x4 pixel block (DSTCD = bpp32) Maximum surface size is 4096x4096 Key information - source and destination base address must be 128-bit word aligned - engine works on FR_BLOCK granularity: transformed surface width in multiples of 16 bytes** transformed surface height in multiples of 16/8/4 lines for bpp8/bpp16/bpp32 FR_BLOCK if surface dimension is not a multiple, software can program FR engine to transform larger surface (round up to next FR_BLOCK in width and height-->transform-->calculate relative memory pointer address) - during a rotational transformation (TRANS_LR, TRANS_RL, ROT_90, ROT_270): the stride of the output surface != the stride of the input surface when working on non-square input input: output: *^---------- --------- *$---------- $ 0 1 2 3 4 5 6 7 ^ 24 16 8 0 ^ ^ ^ ^ | 8 9 10 11 12 13 14 15 | 25 17 9 1 ^ ^ ^ ^ |16 17 18 19 20 21 22 23 | 26 18 10 2 ^ ^ ^ ^ |24 25 26 27 28 29 30 31 | 27 19 11 3 ^ ^ ^ ^ | - - - - * * * * | 28 20 12 4 * * * * | - - - - * * * * | 29 21 13 5 * * * * | - - - - * * * * | 30 22 14 6 * * * * | - - - - * * * * | 31 23 15 7 * * * * - Tiling alignment restrictions subsume FR restrictions because the FR surface base address is the same as the start address. **Register Programming:** FR_MODE - inplace or copy FR_TYPE - type of transformation DSTCD - bpp8, bpp16, bpp32 SRCBA - source base address SRCWIDTH - (width in pixels-1) SRCHEIGHT - (height in lines-1) SRCS - source stride DSTBA - destination base address DSTS - destination stride FR_READWAIT - always set to enable FR inefficiency in the following setup : 1. FR_MODE==SQUARE 2. if(FR_TYPE==YFLIP && SRCHEIGHT==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 Or if(FR_TYPE==XFLIP && SRCWIDTH==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 3. Then, the middle 8 lines/pixels, e.g., YFLIP/16bpp/n==3, line 8 to line 15 will be processed twice. 00 = disable fast rotate - this turns off the second level clock to fr_rotate engine. Please remember to do so! 01 = src/dst copy mode - two separate buffers 10 = square in place - one buffer 11 = blank 0 = DISABLE 1 = SRC_DST_COPY 2 = SQUARE 3 = BLANK
22:21	0x0	CLIPC: Clipping rectangle control, if clip enable, bit 57, XYTDW, should be cleared. 0x=clipping disabled, 10=draw only inside clipping rectangle, 11=draw only outside clipping rectangle
9	0x0	ALPSRCORDST: 32-bit blending mode, output alpha selection 0: source alpha 1: destination alpha 0 = DISABLE 1 = ENABLE
8:4	0x0	ALPTYPE: Alpha blending method FIX: ALPHA blending: Fixed alpha, ALPHA is the value, B5G6R5 * B5G6R5 VCAA: B5G6R5->B5G6R5 PL1BPP: ALPHA blending: Alpha 1 bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero PL2BPP: ALPHA blending: Alpha 2 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero. PL4BPP: ALPHA blending: Alpha 4 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero.

Bit	Reset	Description
		PL8BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1 bit from source B5G5R5A1, alpha (MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4 bits from source A4B4G4R4, alpha (LSB). Dest: B5G6R5. PLS4BPP: ALPHA blending: Alpha 4 bits from source B4G4R4A4, alpha(MSB). Dest: B5G6R5. VCAA: R8G8B8A8->B5G6R5 without reading VCAA plane (surface blit with bpp down convert - implemented in hardware by the VCAA engine; not really a VCAA resolve) PLS8BPP: ALPHA blending: Alpha 8 bits from source/destination (decided by ALPSRCORDST), R8G8B8A8, alpha (MSB). DST: R8G8B8A8 VCAA: R8G8B8A8-> R8G8B8A8, alpha has same blending method as RGB PLS8BX: ALPHA blending: Alpha 8 bits from source B8G8R8A8, alpha (MSB). Dest: B5G6R5. (**Restrictions) VCAA: R8G8B8A8->B5G6R5 PLS1BPPAL: ALPHA blending: Alpha 1 bit from source A1B5G5R5, alpha (LSB). Dest: B5G6R5 VCAA: A1B5G5R5->A1B5G5R5 **Restriction PLS8BX alpha blending has the following restrictions: 1. Source/destination addresses have to be in 128-bit boundary. 2. Destination width has to be a multiple of 4 pixels. 3. Source/Destination strides have to be a multiple of 128 bits. 0 = FIX 1 = PL1BPP 2 = PL2BPP 3 = PL4BPP 4 = PL8BPP 5 = PL44BPP 6 = PLS1BPP 7 = PLS4BPPAL 8 = PLS4BPP 9 = PLS8BPP 10 = PLS8BX 11 = PLS1BPPAL
3	0x0	BEWSWAP: Host port word swap 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 0 = DISABLE 1 = ENABLE

14.2.4 G2SB_G2CONTROLMAIN_0

G2 Blit Formats:

(1) G2 Blit size is based on height in lines, width in pixels, stride in bytes, and pixel size (color depth). Unless alpha blending is enabled, G2 does not care about pixel components.

Source color depths supported by G2 Blit:

- color depth same as destination: G2CONTROLMAIN.SRCCD = 1
- monochrome: G2CONTROLMAIN.SRCCD = 0

(source monochrome color determined by G2SRCBGC, G2SRCFGC)

Pattern color depths supported by G2 Blit:

- color depth same as destination: G2PATOS.PATCD = 1
- monochrome: G2PATOS.PATCD = 0

(pattern monochrome color determined by G2PATBGC, G2PATFGC)

Destination color depths supported by G2 Blit:

- 1 byte per pixel: G2CONTROLMAIN.DSTCD = BPP8
- 2 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP16
- 4 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP32

(2) Alpha blending operates on 3 or 4 component pixels of a specified color depth. One of the components may be alpha (A), depending on the alpha blend format, which is only allowed in certain component positions defined below. The only format rule of the remaining components is that they must be in the same source and destination position because the alpha blend engine does not support component swapping. For example, valid 32BPP formats are: RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888, GRBA_8888->GRBA_8888, etc.

Source formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16:
  - ALPTYPE_PLS1BPP: xxxA_5551 (for example, BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
  - ALPTYPE_PLS4BPP: xxxA_4444 (for example, BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
  - (other): xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32:
  - (all) : xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

Destination formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16: xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

src cd	src format	dst cd	dst format	examples
BPP8	no support	BPP8	no support	---
BPP16	xyzA_5551, xyzA_4444, xyz_565	BPP16	xyz_565	BGRA_5551->BGR_565, RGB_565->RGB565
BPP32	xyzA_8888	BPP32	xyzA_8888	RGBA_8888->RGBA_8888, BGRA_8888-> BGRA_8888 |
BPP32	xyzA_8888	BPP16	xyz_565	RGBA_8888->RGB_565, BGRA_8888-> BGR_565

VCAA Resolve Engine

(1) Format support

srcColor format	srcCoverage format	legal dest format(s)
A1B5G5R5	VCAA	A1B5G5R5
R8G8B8A8	VCAA	R8G8B8A8, B5G6R5
B5G6R5	VCAA	B5G6R5

To program resolve type:

ALPTYPE == FIX: B5G6R5 -> B5G6R5

ALPTYPE == PLS1BPPAL: A1B5G5R5 -> A1B5G5R5

ALPTYPE == PLS8BPP: R8G8B8A8 -> R8G8B8A8

ALPTYPE == PLS8BX: R8G8B8A8 -> B5G6R5

(2) Color surface programming

- color surface base address: SRCBA
- sub-surface start x: SRCX (pixel index)
- sub-surface start y: SRCY (line index)
- color surface stride: SRCS (bytes)

The engine supports sub-surface resolve. One can imagine a full color buffer of 1024x768, but

```
    [pseudo_code]
      color_surface_start_address_fetch = pcurctx->regs.rG2SRCBA.uSRCBA() +
       (pcurctx->regs.rG2SRCPS.uSRCX() * VCAAState.color_surface_depth) +
       (pcurctx->regs.rG2SRCPS.uSRCY() * VCAAState.color_surface_stride);
```

(3) VCAA surface programming

- vcaa surface base address: PATBA
- vcaa surface stride: PATST (bytes)

```
[pseudo_code]
     vcaa_surface_start_address_fetch = pcurctx->regs.rG2PATBA.uPATBA();
```

Software *MUST* directly program this register to the proper sub-surface location in the VCAA surface which corresponds to the SRCX and SRCY programming. The calculation is PATBA = vcaa_base_address + SRCX * (1) + SRCY * PATST;

(4) Resolve surface programming

- resolve window width: DSTWIDTH (pixels)
- resolve window height: DSTHEIGHT (lines)
- resolve output stride: DSTS (bytes)

These values program the width and height of the resolved surface. In the base case,

- DSTWIDTH = color surface width
- DSTHEIGHT = color sufrace height
- DSTS = resolve surface format bpp * DSTWIDTH

The VCAA engine technically supports DSTX and DSTY (like SRCX and SRCY).

```
 [pseudo_code]
    resolve_surface_start_address_put = pcurctx->regs.rG2DSTBA.uDSTBA() +
      (pcurctx->regs.rG2DSTPS.uDSTX() * resolve_surface_color_depth) +
      (pcurctx->regs.rG2DSTPS.uDSTY() * resolve_surface_color_stride);
```

(5) Resolving a pixel

The coverage surface is a 8 bits per pixel surface (C4X4). The surface is initialized to all 1s by 3d.

For a given pixel, we need to look up 4 bits of coverage data. If the data is 0xf, the pixel does not need to be resolved. Otherwise, we need to reblend the pixel with certain weightings of its neighbors (i.e., we're on the edge of some geometry)

If any coverage bit is 0, we need to resolve the color. To do so, we calculate the following equation:

```
   color_new = (
```

```
                20*color_old +
                27*(
                    ((cover_down  == 0) ? color_down  : color_old) +
                    ((cover_right == 0) ? color_right : color_old) +
                    ((cover_left  == 0) ? color_left  : color_old) +
                    ((cover_up    == 0) ? color_up    : color_old)
                    )
            ) / 128;
```

If coverage bit == 0, then we use the neighbor color value in the resolve

If coverage bit == 1, then we use the center color value in the resolve

The resolve is done on a per channel (r/g/b/a) basis:

- format convert input channel to 8-bit
- perform resolve equation above
- format convert to output format

For the resolution of the alpha channel:

- In B5G6R5: no alpha information. do nothing.
- In A1B5G5R5: no resolve. keep the source color alpha bit.
- In R8G8B8A8: normal 8-bit channel resolve

(6) Resolving a pixel on the edge

When on an edge, the vcaa engine treats the coverage bit as a 1. Use the center color value. This holds true even on the edge of the sub-surface resolve window when located in the middle of the larger surface

(7) Maximum dimensions

The maximum resolve sub-surface is 4096 pixels wide.

(8) Surface restrictions

The VCAA engine has the following restrictions with respect to input surfaces

- input base address is 128-bit memory word aligned

- input stride is a multiple of 128 bits (16 bytes)

The VCAA engine has the following restrictions with respect to output surfaces

- output base address is 128-bit memory word aligned

- output stride is a multiple of 128 bits (16 bytes)

(9) Coverage surface layout

The surface is C4X4.

Surface layout (C4X4)

```
       (lsb2msb)  C0_C1_C2_C3_X_X_X_X     (X is a don't care.  Qrast inits to 1)
       (msb2lsb)  X_X_X_X_C3_C2_C1_C0


    +          o          o
         C0
            C1    Cov(0,1)
     C2
```

```
          C3
     o          o          o


       Cov(1,0)    Cov(1,1)


     o          o          o


     +/o = pixel centers
```

In the example above,

```
        Cov(0,0) = X_X_X_X_C3_C2_C1_C0
        C3       = V_C3(Cov(0,0));
        ..         ..
        C0       = V_C0(Cov(0,0));
     #define V_C3(bits) (bits & 0x8)
     #define V_C2(bits) (bits & 0x4)
     #define V_C1(bits) (bits & 0x2)
     #define V_C0(bits) (bits & 0x1)
```

Resolving a pixel

In an ideal world, the coverage would be simply laid out such that when resolving Pixel(x,y), we would need only to fetch Cov(x,y) to get all 4 bits of information.

In actuality, the coverage surface was rearranged to help qrast in performance. To resolve Pixel(x,y) gr2d has to fetch 4 C4X4 bytes [Cov(x,y), Cov(x-1,y), Cov(x,y-1), Cov(x-1,y-1)]. From each C4X4 byte, it extracts one bit which is used in the resolve equations shown in (5).

Need the logical VCAA bits (DRLU) for blend which is spread across four C4X4 coverage bytes

```
          U
    L
        o
            R
      D
```

where

```
     o = Pixel(x,y)
     D = V_C1(Cov(x-1,y))
     R = V_C0(Cov(x,y))
     L = V_C3(Cov(x-1,y-1))
     U = V_C2(Cov(x,y-1))
```

Offset: 0x1f | Byte Offset: 0x7c | Read/Write: R/W | Reset: 0x00000000 (0bx000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: Pattern Data Select: SRCSEL and PATSEL cannot be both enabled.

Bit	Reset	Description
		0 = SCREEN 1 = MEMORY
28	0x0	DSTDIR: destination direct addressing
27	0x0	SRCDIR: source direct addressing
26:25	0x0	GCSW: Display Switching Window Control (GCSW[1:0]). This parameter controls multi-buffering for Display. x0: At end of current command, do not send signal to Display to switch buffer. 01: two buffers, dstba and dstba_b are used 11: three buffers, dstba, dstba_b, dstba_c are used
24	X	SRCBAS: Source base address select: 0=srcba, 1=dstba. This is not used for Line Draw and if source data comes from host memory.
23:22	0x0	SRCT: Source transparency enable: 0x=source transparency disabled, 10=mono source background transparency or color source transparency, 11=mono source foreground transparency or inverse color source transparency. NOTE: When source transparency is enabled and SRCCD==0(mono) SRCBGC!=SRCFGC should be satisfied.
21	X	HLMONO: Start from Msb or lsb in byte when mono expansion If HLMONO is 1, bit 0 (the lsb) is the first bit. If HLMONO is 0, bit 7 (the msb) is the first bit
20	X	SRCCD: 0: Source mono 1: Source has same color depth as destination SRCCD==0 xdir/ydir has to be 0
19:18	0x0	DSTT: Destination read transparency enable: 0x=destination read transparency disabled 10=color destination read transparency 11=inverse color destination read transparency. 20 rw INIC Initiate Command (1=initiate command immediately, 0=wait for launch write)
17:16	X	DSTCD: Destination color depth: 00=8-bpp 01=16-bpp 10=32-bpp 11=reserved 0 = BPP8 1 = BPP16 2 = BPP32 3 = RESERVED3
15	0x0	SRCSEL: Source Data Select: 0 = SCREEN 1 = MEMORY
14	0x0	YFLIP: flip y direction to make image upside down or the other way. If YFLIP==1, ROP cannot include destination. 0 = DISABLE 1 = ENABLE
13	0x0	PATPACK: Pattern Data is in Pack Mode. PATLNGAP in G2PATPACK is the line gap for pattern packed mode If (PATPACK && ~PATSEl), pattern data is packed and from screen, PATMONOW/PATMONOH should be programmed properly to fetch pattern data

Bit	Reset	Description
		from the frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
12	0x0	SRCPACK: Source Data is in Pack Mode. SRCLNGAP in G2SRCPACK is the line gap for source packed mode. If (SRCPACK && ~SRCSEL), source data is packed and from screen, SRCMONOW/SRCMONOH should be programmed properly to fetch data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
11	0x0	XYTDW: xy transpose, Line stride DSTS has to be 16-byte aligned if enabled. If XYTDW==1, ROP cannot include destination.
10	0x0	YDIR: 0=incrementing, 1=decrementing. ydir should not be set when source surface has different color depth as destination surface. SRCCD==0 (mono src) OR PATCD==0 (mono pat), ydir has to be 0
9	0x0	XDIR: 0=incrementing, 1=decrementing. xdir should not be set when source surface has different color depth as destination surface. For example, ALPTYPE=PLS8BX (32bpp blending with 16bpp), xdir has to be 0 SRCCD==0 (mono src) OR PATCD==0 (mono pat), xdir has to be 0

8	0x0	PATFL: When mono pattern is set, we use mono tile pattern fill.
		Current pattern client can support a 16x16 bit tile which can be used to generate a larger surface (i.e., tile this surface in the x and y direction). The tile is stored in memory. Currently, PATXO and PATYO indicate where to start in the 16x16 tile when expanding the monochrome data. See patxo and patyo comments below for programming.
		When xdir==1 Patxo = pattern width - (patxo+destination width)&0xF -- For xoffsets y offsets PATYO remains the same when xdir/ydir change.
		How the tile replication pattern looks based on xdir/ydir (without programming patxo above): xdir=0 ydir=0 xdir=1 ydir=0 +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ // xdir=0 ydir=1 xdir=1 ydir=1 Mono tile is +---+ |^|^| |~~~| +---+
		0 = DISABLE
		1 = ENABLE

7	0x0	PATSLD: BitBlt Solid Pattern Fill: 1=enable. BGC will be used as the color value. 0 = DISABLE 1 = ENABLE
6	0x0	SRCSLD: BitBlt Solid Source Color Fill: 1=enable. FGC will be used as the color value. 0 = DISABLE 1 = ENABLE
5	0x0	ALPEN: BitBlt Alpha Blending, 1=enable. 0=disable, when both Faden and alpen are 1, output=Source*alpha_v + fadoff, alpha_v is decided by alptype
4	0x0	FADEN: BitBlt Source Copy Fade enable, 1=enable (share with mltln), only support 16bpp mode 0 = DISABLE 1 = ENABLE
3	0x0	TEST0BIT: Command finish timing bit. 0: 2D command finishes when last data has been pushed to memory write client. 1: 2D command waits memory write client to be idle to finish. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	TURBOFILL: fast fill rectangle in 128 bits/clock. Some limitations with this mode:srcsld==1 rop==0xcc, no clipping, no transparency xdir==0, ydir==0, flip==0, xytdw==0 Results are undefined if the above limitations are not satisfied.
1:0	0x0	CMDT: Command Type: 00=BitBlt 01=Line Draw 10=VCAA 11=reserved When the raise command is executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

14.2.5 G2SB_G2ROPFADE_0

Offset: 0x20 | Byte Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:24	X	FADOFF:
23:16	X	FADCOE:
7:0	0x0	ROP: If YFLIP==1 or XYTDW==1, ROP cannot include destination. Since destination may have been corrupted before reading out.

14.2.6 G2SB_G2ALPHABLEND_0

Offset: 0x21 | Byte Offset: 0x84 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
31:24	X	ALPHA1V
23:16	X	ALPHA0V
8	0x0	ALPHAINV
7:0	X	ALPHA

14.2.7 G2SB_G2CLIPLEFTTOP_0

ClipRect Methods. Top-left bounds are inclusive.

Offset: 0x22 | Byte Offset: 0x88 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPT
14:0	X	CLIPL

14.2.8 G2SB_G2CLIPRIGHTBOT_0

Bottom-right bounds are exclusive

Offset: 0x23 | Byte Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPB
14:0	X	CLIPR

14.2.9 G2SB_G2PATPACK_0

Pattern Packed Mode

Pattern methods

G2PATPACK should be used to specify the line gap. Use G2PATPACK_SIZE to program height and width. PACK is generally only useful with narrow monochrome surfaces

Note: PACK is not officially supported when the surface is tiled.

Offset: 0x24 | Byte Offset: 0x90 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

14.2.10 G2SB_G2PATPACK_SIZE_0

Pattern packed mode

Packed mode, pattern data line gap. Byte

G2PATPACK_SIZE -- extension of G2PATPACK, this register only holds the size of the packed data

Note: PACK is not officially supported when the surface is tiled.

Offset: 0x25 | Byte Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	PATMONOH: Pattern mono data height in packed mode, byte
15:0	X	PATMONOW: Mono data width in packed mode, byte

14.2.11 G2SB_G2PATBA_0

Offset: 0x26 | Byte Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBA: If (PATFL==1){ It has to be 16-byte aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

14.2.12 G2SB_G2PATOS_0

Offset: 0x27 | Byte Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxx00xxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	PATYO: y offset for mono tile pattern fill. see PATFL
27:24	X	PATXO: x offset for mono tile pattern fill. see PATFL
22:21	0x0	PATT: Mono pattern transparency enable: 0x=pattern transparency disabled, 10=mono pattern background transparency or color pattern transparency, 11=mono pattern foreground transparency or inverse color pattern transparency. NOTE: When pattern transparency is enabled and PATCD==0 (mono) PATBGC!=PATFGC should be satisfied.
16	X	PATCD:0 mono1 same as dstcd PATCD==0 xdir/ydir has to be 0
15:0	X	PATST: stride

14.2.13 G2SB_G2PATBGC_0

Offset: 0x28 | Byte Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

14.2.14 G2SB_G2PATFGC_0

Offset: 0x29 | Byte Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

14.2.15 G2SB_G2PATKEY_0

Offset: 0x2a | Byte Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

14.2.16 G2SB_G2DSTBA_0

Offset: 0x2b | Byte Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA: Destination base address (byte address)

14.2.17 G2SB_G2DSTBA_B_0

Offset: 0x2c │ Byte Offset: 0xb0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_B: Destination base address (byte address) only usable in hardware trigger mode by enable gcsw

14.2.18 G2SB_G2DSTBA_C_0

Offset: 0x2d │ Byte Offset: 0xb4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_C: Destination base address (byte address)

14.2.19 G2SB_G2DSTST_0

Offset: 0x2e │ Byte Offset: 0xb8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

14.2.20 G2SB_G2SRCPACK_0

Surface methods: G2SRCPACK should only be used to specify the line gap. Use G2SRCPACK_SIZE to program height and width.

Source Data Packed Mode

Offset: 0x2f │ Byte Offset: 0xbc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

14.2.21 G2SB_G2SRCPACK_SIZE_0

In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, the required width/stride is 16 byte aligned. Packed mode - source mono data line gap

G2SRCPACKS_SIZE -- extension of G2SRCPACK, this register only holds the size of the packed data

Source Data Packed Mode

Offset: 0x30 │ Byte Offset: 0xc0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCMONOH: Packed mode - source mono data height in bytes
15:0	X	SRCMONOW: Packed mode - source mono data width in bytes

14.2.22 G2SB_G2SRCBA_0

SRC Base Address

Offset: 0x31 | Byte Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA: Source base address (byte address)

14.2.23 G2SB_G2SRCBA_B_0

SB only

Offset: 0x32 | Byte Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA_B: This parameter specifies the start address of source image stored in the image buffer memory. In 4:2:0 format mode, this image block accommodates for Y-image. This address specifies byte position; however, bits [2:0] are restricted with respect to the data formats to fit multiple pixels in one memory word (8 bytes). For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format. Since one Y pixel takes 8 bits, all 8 byte positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there are no restrictions on this value.)

14.2.24 G2SB_G2SRCST_0

Offset: 0x33 | Byte Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS: Source Stride coordinate (bytes) with respect to SRCBA. In order to fit multiple pixels in one memory word (8 bytes), bits [2:0] are restricted with respect to the data formats. For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format.

14.2.25 G2SB_G2SRCBGC_0

srcColors

Offset: 0x34 | Byte Offset: 0xd0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

14.2.26 G2SB_G2SRCFGC_0

Offset: 0x35 | Byte Offset: 0xd4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

14.2.27 G2SB_G2SRCKEY_0

Offset: 0x36 | Byte Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

14.2.28 G2SB_G2SRCSIZE_0

Offset: 0x37 | Byte Offset: 0xdc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	SRCHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	SRCWIDTH

14.2.29 G2SB_G2DSTSIZE_0

Offset: 0x38 | Byte Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	DSTHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	DSTWIDTH: In 2D mode, the largest number can be programmed is 0x7fef

14.2.30 G2SB_G2SRCPS_0

ImageBlit Methods

Offset: 0x39 | Byte Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCY
15:0	X	SRCX: SRCX[2:0] are ignored in SRCCD==0 (mono expansion). The first bit of the first byte (bit7 if HLMONO==0, or bit0 if HLMONO==1) always expand to DSTX,DSTY

14.2.31 G2SB_G2DSTPS_0

Offset: 0x3a | Byte Offset: 0xe8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	DSTY
15:0	X	DSTX: NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero

14.2.32 G2SB_G2CBDES_0

Circular Buffer

Offset: 0x3b | Byte Offset: 0xec | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TOPCLIP: Top clipping at the first buffer, (buffer start) refer to TOP_CBLINE in G2CBDES2 0= disable 0 = DISABLE 1 = ENABLE
30:16	X	CBLINE: Vertical line number in one buffer
7:0	X	CBCOUNT: This specifies the number of buffers in circular buffer feature

14.2.33 G2SB_G2CBSTRIDE_0

Offset: 0x3c | Byte Offset: 0xf0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	X	CBUVSTRIDE: Chroma Buffer Stride default is half of luma 00: 1/2 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 2 bytes. 01: Equal to Luma Buffer Stride 10: 1/4 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 4 bytes. 1x: Reserved 0 = CBS2X 1 = CBS1X 2 = CBS4X
23:0	X	CBSTRIDE: Video Buffer Luma(or RGB) Buffer Stride. This is luma buffer stride (in bytes)

14.2.34 G2SB_G2LINESETTING_0

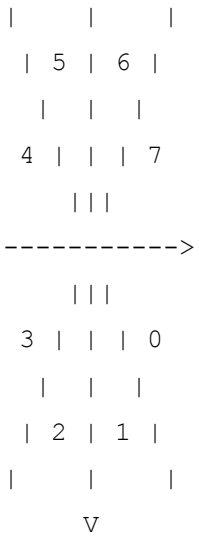
Line Methods

Offset: 0x3d | Byte Offset: 0xf4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	OCTANTS: 000: octant 0 001: octant 1 010: octant 2 011: octant 3 100: octant 4 101: octant 5 110: octant 6 111: octant 7
28	X	LINEUSEOCTANT: use OCTANTS in G2LINEDELTAN register instead of MAJOR LINEXDIR LINEYDIR
27	X	DROPLASTP: Draw last pixel or not
26	X	LINEYDIR

Bit	Reset	Description
25	X	LINEXDIR
24	X	MAJOR: 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

14.2.35 G2SB_G2LINEDELTAN_0



Offset: 0x3e | Byte Offset: 0xf8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

14.2.36 G2SB_G2LINEDELTAM_0

Offset: 0x3f | Byte Offset: 0xfc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

14.2.37 G2SB_G2LINEPOS_0

Offset: 0x40 | Byte Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LINEYPOS
15:0	X	LINEXPOS

14.2.38 G2SB_G2LINELEN_0

Offset: 0x41 | Byte Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

14.2.39 G2SB_G2CSCFOURTH_0

Offset: 0x42 | Byte Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:16	X	G2V: Multiplier for G for V generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.368 (decimal) or 0x12F. For any other combination, this parameter is ignored.
8:0	X	G2U: Multiplier for G for U generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.291 (decimal) or 0x125. For any other combination, this parameter is ignored.

14.2.40 G2SB_G2SRCST_B_0

Offset: 0x43 | Byte Offset: 0x10c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

14.2.41 G2SB_G2UVSTRIDE_0

Offset: 0x44 | Byte Offset: 0x110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

14.2.42 G2SB_G2CBDES2_0

Circular Buffer Controller 2

Offset: 0x45 | Byte Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

14.2.43 G2SB_G2TILEMODE_0

Offset: 0x46 | Byte Offset: 0x118 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx0xxx0xxx0xxx0xxx0xxx0)

Bit	Reset	Description
20	0x0	DST_WR_TILE_MODE: Destination surface 0 = LINEAR 1 = TILED
16	0x0	DST_RD_TILE_MODE: Same as destination write unless DST_RD_WR_SEP (not supported) 0 = LINEAR

Bit	Reset	Description
		1 = TILED
12	0x0	PAT_UV_TILE_MODE:UNUSED 0 = LINEAR 1 = TILED
8	0x0	PAT_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED
4	0x0	SRC_UV_TILE_MODE:UV surface, ignored in RGB mode 0 = LINEAR 1 = TILED
0	0x0	SRC_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED

14.2.44 G2SB_G2PATBASE_0

Offset: 0x47 │ Byte Offset: 0x11c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PAT_BASE: Pattern base address in tile mode, PATBA is the linear address where pixel start

14.2.45 G2SB_G2SRCBA_SB_SURFBASE_0

SB_SURFBASE Registers

These registers need only be programmed when using SB. They point to the base of the various source and destination surfaces. Technically, they are only needed when tiling is enabled, but there is no harm in always programming them.

Their counterpart registers (_SB_SURFBASE with SB_SURFBASE stripped off) indicate the location of the first pixel to be sourced or written within the surface.

This register exists to mimic the X, Y (SRCX, SRCY) functionality found in the BitBlt engine.

For example, to get at pixel X, Y:

SRCBA = SRCBA_SB_SURFBASE + Y*stride + X*Bpp

Offset: 0x48 │ Byte Offset: 0x120 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRC_ADDR: Surface address corresponding to G2SRCBA: -base of interleaved sources (RGB, YUV) -base of Y plane. Only used by the StretchBlit Engine

14.2.46 G2SB_G2DSTBA_SB_SURFBASE_0

Offset: 0x49 │ Byte Offset: 0x124 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_ADDR: Surface address corresponding to G2DSTBA. Only used by the StretchBlit Engine

14.2.47 G2SB_G2DSTBA_B_SB_SURFBASE_0

Offset: 0x4a | Byte Offset: 0x128 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_B_ADDR: Surface address corresponding to G2DSTBA_B. Only used by the StretchBlit Engine, and G2CONTROLSB.DBSEL() is enabled

14.2.48 G2SB_G2VBA_A_SB_SURFBASE_0

Offset: 0x4b | Byte Offset: 0x12c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	V_ADDR: Surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of V plane. Only used by the StretchBlit Engine

14.2.49 G2SB_G2UBA_A_SB_SURFBASE_0

Offset: 0x4c | Byte Offset: 0x130 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	U_ADDR: Surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of U plane. Only used by the StretchBlit Engine

14.3 G2SB CTX1 Registers

14.3.1 G2SB_CTX1_INCR_SYNCPT_0

Memory Controller Tiling Definitions

Offset: 0x4000 | Byte Offset: 0x10000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	INDX: syncpt index value

14.3.2 G2SB_CTX1_INCR_SYNCPT_CNTRL_0

Offset: 0x4001 | Byte Offset: 0x10004 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

14.3.3 G2SB_CTX1_INCR_SYNCPT_ERROR_0

Offset: 0x4002 | Byte Offset: 0x10008 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

14.3.4 G2SB_CTX1_G2CLASSCHANNEL_REGONLY_0

This register can be accessed by both register read/write and command FIFO in order not to break exsting code , keep suffix REGONLY.

2D syncpt behavior

- Each 2D context has its own syncpt client.
- OP_DONE increments when 2D's memory write client is idle, meaning all preceding operations are complete.
- RD_DONE behaves like OP_DONE
- REG_WR_SAFE behaves like IMMEDIATE meaning a syncpt increment returns immediately and does not indicate that a register write will NOT block. The implications are that 2D's host FIFO can fill and block access to 2D, which will indeterminately block a direct CPU access.

Offset: 0x4008 | Byte Offset: 0x10020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

14.3.5 G2SB_CTX1_G2TRIGGER_0

Control methods, since they are not programmed often, put them in the beginning

Offset: 0x4009 | Byte Offset: 0x10024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

14.3.6 G2SB_CTX1_G2TRIGGER1_0

Offset: 0x400a | Byte Offset: 0x10028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

14.3.7 G2SB_CTX1_G2TRIGGER2_0

Offset: 0x400b | Byte Offset: 0x1002c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

14.3.8 G2SB_CTX1_G2CMDSEL_0

Offset: 0x400c | Byte Offset: 0x10030 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxx0xxxxxxxx00x0x0000000xxxx)

Bit	R/W	Reset	Description
28	RW	0x0	PRIORITY: Indicates priority of the context, note vi-triggered contexts take priority over HIGH 0 = LOW 1 = HIGH
27:25	RW	X	LINKVAL
24	RW	0x0	LINKEN: Test purpose
23:16	RW	X	BUFFER_INDEX
15	RW	0x0	FRAME_END
14	RW	0x0	FRAME_START: host writes this bit to tell which buffer is ready
10	RW	0x0	CLIP_SOURCE_TOP_BOTTOM: Reserved for hardware test. Software should not use this bit. 0 = DISABLE 1 = ENABLE
9:8	RW	0x0	G2OUTPUT: 00: to memory. 01: to EPP 0 = MEMORY 1 = EPP 2 = RESERVED2 3 = RESERVED3
7	RW	0x0	CBSBDISABLE
6	RW	0x0	HOSTTRIGGER: host trigger the command, host also need program circular buffer related data
5	RW	0x0	VITRIGGER:VI trigger enable, can be circular buffer or full frame
4	RW	0x0	CBENABLE: circular buffer feature enable. 0= disable 0 = DISABLE 1 = ENABLE
0	RO	X	SBOR2D: 0 = G2 1 = SB

14.3.9 G2SB_CTX1_G2RAISE_0

Offset: 0x400d │ Byte Offset: 0x10034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE: Push back to read FIFO when all commands in the channel are done.

14.3.10 G2SB_CTX1_G2HOSTSET_0

Offset: 0x400f │ Byte Offset: 0x1003c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	HSTFPXL: specifies the first pixel location (in byte) in the first source data entry. In mono mode pixel should always start on a byte boundary
3:0	0x0	HSTLNGAP: specifies distance (in byte) from last pixel of a line to the first pixel of next line. In mono mode, the value is the nearest byte number. For example, if the gap is 3 bits, HSTLNGAP should be 1 (byte), if the gap is 9 bits, HSTLNGAP should be 2 (byte)

14.3.11 G2SB_CTX1_G2HOSTFIFO_0

Offset: 0x4010 │ Byte Offset: 0x10040 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

14.3.12 G2SB_CTX1_G2VDDA_0

Offset: 0x4011 │ Byte Offset: 0x10044 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VDSTEP: Vertical Scaling DDA. Reset value: xxxx-xxxxh. Vertical DDA Step (Increment) Value (VDSTEP[18:0]) (upper 13 bits should be set to '0'). This parameter specifies the increment value of the DDA used for vertical scaling and it is in the form of a 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_height-1-VDTINI) / (Actual_destination_height-1) Truncate the remaining bits to keep the 12-bit fraction. Since we have to meet (Actual_source_height-1)*1.0 >= (Actual_destination_height - 1)*VDSTEP + VDTINI. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half line spacing for up to approximately 2000 target lines. For example, image expansion from 240 lines to 400 lines VDSTEP[17:0] = 19'b00_0000_1001_1001_1010 and image contraction from 240 lines to 150 lines VDSTEP[17:0] = 19'b00_0001_1001_1001_1010.

14.3.13 G2SB_CTX1_G2VDDAINI_0

Offset: 0x4012 │ Byte Offset: 0x10048 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	VDBINI: Vertical DDA Initial Value for Bottom-Field (VDBINI[7:0]) (see description above)
7:0	X	VDTINI: Vertical Scaling DDA Initial Values. Vertical DDA Initial Value for Top-Field (VDTINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for vertical scaling. Given a pair of consecutive source image lines representing positions 0.0 and 1.0, a destination line-image will be created at any position in [0.0, 1.0), that is specified by the fraction part of the

Bit	Reset	Description
		vertical DDA. The 8-bit initial fraction value specifies the position of the first output (destination) line-image located between the first and the second input (source) line-images, respectively at position 0.0 and 1.0. This parameter provides a way to compensate relative displacement of top and bottom fields of a source image. Suppose, 2-to-1 interlaced original video field images #1 and #2 are in the image buffer, to be displayed in an overlay window. Assume that the #1 field image is the top-field. Using VDTINI = 8'hC0 for the top-field image and VDBINI = 8'h40 for the bottom-field image, for example, the two images from #1 and #2 fields get mapped to identically positioned destination images in the overlay window.

14.3.14 G2SB_CTX1_G2HDDA_0

Offset: 0x4013 | Byte Offset: 0x1004c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HDSTEP: Horizontal Scaling DDA. Reset value: xxxx-xxxxh. Horizontal DDA Step (Increment) Value (HDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for horizontal scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_width-1-HDINI) / (Actual_destination_width-1) Truncate the remaining bits to keep the 12-bit fraction. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half pixel-spacing for up to approximately 2000 destination pixels. For example, image expansion from 720 pixels to 800 pixels HDSTEP[17:0] = 19'b000_0000_1110_0110_0111 and image contraction from 720 pixels to 250 pixels HDSTEP[17:0] = 19'b000_0010_1110_0001_0101.

14.3.15 G2SB_CTX1_G2HDDAINILS_0

Offset: 0x4014 | Byte Offset: 0x10050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	HDINI: Horizontal Scaling DDA Initial Value. Horizontal DDA Initial Value (HDINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for horizontal scaling. Given a group of six consecutive source pixels that the two pixels at the center representing positions 0.0 and 1.0, a destination pixel will be created at any position in [0.0, 1.0), that is specified by the fraction part of the horizontal DDA. The 8-bit initial fraction value specifies the position of the first output (destination) pixel located between the first and the second input (source) pixels, respectively at position 0.0 and 1.0.For horizontal scaling, this value may normally be set to 0.

14.3.16 G2SB_CTX1_G2CSCFIRST_0

The next 3 registers contain 8 parameters used by the YCbCr (YUV) to RGB color space conversion logic. This logic can also be used for RGB gain/gamma correction in RGB to RGB modes, and for luma gain in YUV to YUV modes.

Two parameters (G2U, G2V) have been added for RGB to YUV conversion in address 0x38. The equations have the form:

if (yuv2rgb)

(r,g,b) = matrix*(y+sbreg_yos,u,v)

matrix: sbreg_cyx, sbreg_cur, sbreg_cvr,
sbreg_cyx, sbreg_cug, sbreg_cvg,
sbreg_cyx, sbreg_cub, sbreg_cvb,

else if (rgb2rgb)

(r,g,b) = matrix*(r,g+sbreg_yos,b)

matrix:	sbreg_cvr,	0,	sbreg_cur,
	sbreg_cvg,	sbreg_cyx,	sbreg_cug,
	sbreg_cvb,	0,	sbreg_cub,

else if (yuv2yuv)

(y,u,v) = matrix*(y,u,v)

matrix:	sbreg_cub,	0,	sbreg_cvb,
	sbreg_cug,	sbreg_cyx,	sbreg_cvg,
	sbreg_cur,	0,	sbreg_cvr,

else if (rgb2yuv)

(y,u,v) = matrix*(r,g,b) + (sbreg_yos,0,0)

matrix:	sbreg_cvb,	sbreg_cyx,	sbreg_cub,
	sbreg_cvg,	sbreg_g2u,	sbreg_cug,
	sbreg_cvr,	sbreg_g2v,	sbreg_cur,

Offset: 0x4015 │ Byte Offset: 0x10054 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	YOS: Y-Offset (YOFFSET[7:0]) for YUV generation. This parameter consists of 8-bit 2's complement in the range [-128,127]. For YUV->RGB, the recommended value is -16 (decimal) or 0xF0. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is +16 (decimal) or 0x10.
21:12	X	CVR: multiplier for V/R for V or R generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 1.5960 (decimal) or 0x0CCIf source data is in RGB format, this parameter may be used as gain adjustment for R component. For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038.
9:0	X	CUB: multiplier for U/B for Y or B generation. Consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 2.0172 (decimal) or 0x102. If source data is in RGB format, this parameter may be used as gain adjustment for B component. This register changes precision when doing RGB to YUV conversion (SIFMT=1xxx, DIFMT=0xxx). CUB becomes s1.8 and the recommended value is +0.098 or 0x019.

14.3.17 G2SB_CTX1_G2CSCSECOND_0

Offset: 0x4016 │ Byte Offset: 0x10058 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	CYX: Multiplier for Y/G (G gain). This positive-only parameter consists of 8-bit magnitude (1.7). For YUV->YUV, the recommended value is 1.1644 (decimal) or 0x95. For YUV->YUV or RGB->RGB, this may be used as gain adjustment for Y or G component. For RGB->YUV, the recommended value is +0.504 (decimal) or 0x041.
21:12	X	CUR: Multiplier for U/B for V or R generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This parameter takes non-zero value if hue is rotated. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.071 (decimal) or 0x209.
8:0	X	CUG: Multiplier for U/B for U or G generation. Consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB, the recommended value is -0.3918 (decimal) or 0x132. For RGB->RGB, this parameter should be set to 0. For YUV->YUV, this parameter should be set to 1 (0x080). For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038.

14.3.18 G2SB_CTX1_G2CSCTHIRD_0

Offset: 0x4017 │ Byte Offset: 0x1005c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	CVB: Multiplier for V/R for Y or B generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This coefficient takes non-zero value if hue is rotated. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is +0.257 (decimal) or 0x021.
8:0	X	CVG: Multiplier for V/R for U or G generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB the recommended value is -0.8130 (decimal) or 0x168. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.148 (decimal) or 0x113.

14.3.19 G2SB_CTX1_G2CMKEYL_0

When key generation is enabled, the value of this register is the lower color/chroma limit and G2CMKEYU is the upper color/chroma limit for component pixel data. Three component signal values (YCrCb/YUV or RGB) of every output pixel are compared to a set of three ranges that are specified by three pairs of lower and upper color/chroma key values.

If the Key Polarity is 0, the Key is set to 1 only when all three component values of an input pixel are respectively within the set of ranges (inclusive of the limit values), else the Key is set to 0.

If the Key Polarity is 1, the Key is set to 1 when any of the three component values of an input pixel are outside the set of ranges; otherwise, the Key is set to 0.

StretchBLT, Color/Chroma Key Lower Limit Register

Offset: 0x4018 │ Byte Offset: 0x10060 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRL:R or Cr Color Chroma Key Lower Limit (CKRL[7:0]) Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGL:G or Cb Color Chroma Key Lower Limit (CKGL[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
7:0	X	CKBL: B or Y Color/Chroma Key Lower Limit (CKBL[7:0])

14.3.20 G2SB_CTX1_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0x4019 │ Byte Offset: 0x10064 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRU: R or Cr Color Chroma Key Upper Limit (CKRU[7:0]) Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGU: G or Cb Color/Chroma Key Upper Limit (CKGU[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity the minimum signal level to the maximum signal level.
7:0	X	CKBU: This is B or Y color/chroma key upper limit value.

14.3.21 G2SB_CTX1_G2UBA_A_0

StretchBLT, Source Image Data U base address A-buffer

Offset: 0x401a │ Byte Offset: 0x10068 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SU1SA: Start Address of Source U-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.3.22 G2SB_CTX1_G2VBA_A_0

Offset: 0x401b │ Byte Offset: 0x1006c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SV1SA: Start Address of Source V-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.3.23 G2SB_CTX1_G2SBFORMAT_0

Offset: 0x401c │ Byte Offset: 0x10070 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12:8	X	DIFMT: Destination Image Data Format This parameter defines the data format of destination output. There are two groups of data formats, RGB and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement. 00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 01000: bpp16 5-6-5 {R[4:0], G[5:0], B[4:0]} 01001: RESERVED 01010: RESERVED 01011: RESERVED 01100: bpp16 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]} 01101: RESERVED 01110: R8G8B8 A801111 B8G8R8A8 1xxxx: RESERVED 0 = U8Y8V8Y8_OB 1 = Y8U8Y8V8_OB 2 = Y8V8Y8U8_OB 3 = V8Y8U8Y8_OB 4 = U8Y8V8Y8_TC 5 = Y8U8Y8V8_TC 6 = Y8V8Y8U8_TC 7 = V8Y8U8Y8_TC 8 = B5G6R5 9 = RESERVED9 10 = RESERVED10

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>11 = RESERVED11
12 = B5G6R5BS
13 = RESERVED13
14 = R8G8B8A8
15 = B8G8R8A8
16 = RESERVED16
17 = RESERVED17
18 = RESERVED18
19 = RESERVED19
20 = RESERVED20
21 = RESERVED21
22 = RESERVED22
23 = RESERVED23
24 = RESERVED24
25 = RESERVED25
26 = RESERVED26
27 = RESERVED27
28 = RESERVED28
29 = RESERVED29
30 = RESERVED30
31 = RESERVED31</td></tr>
<tr><td>4:0</td><td>X</td><td>SIFMT: This parameter defines the data format of source input.
There are two groups of data formats, RGB format and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement.
00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
01000: B5G6R5 5-6-5 {R[4:0], G[5:0], B[4:0]}
01001: RESERVED
01010: RESERVED
01011: RESERVED
01100: B5G6R5 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]}
01101: RESERVED
01110: R8G8B8A8
01111: B8G8R8A8
1xxxx: RESERVED
StretchBlit Inputs: RGB inputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV420 input = {YUV420 is converted into 4:2:2 UYVY via memory client} YUV422 inputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}
StretchBlit Outputs: RGB outputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV422 outputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}
StretchBlit Input/Output Rules
<table>
<tr><th>src format</th><th>internal sb format</th><th>dst format</th></tr>
<tr><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td><td>R8G8B8A8</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*,</td><td>YUV 4:4:4</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
</table></td></tr>
</table>

Bit	Reset	Description
		V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420 | | StretchBlit color space converter supports YUV->RGB, YUV->YUV (gain) and RGB->RGB (gain). There is no support for RGB->YUV. 0 = U8Y8V8Y8_OB 1 = Y8U8Y8V8_OB 2 = Y8V8Y8U8_OB 3 = V8Y8U8Y8_OB 4 = U8Y8V8Y8_TC 5 = Y8U8Y8V8_TC 6 = Y8V8Y8U8_TC 7 = V8Y8U8Y8_TC 8 = B5G6R5 9 = RESERVED9 10 = RESERVED10 11 = RESERVED11 12 = B5G6R5BS 13 = RESERVED13 14 = R8G8B8A8 15 = B8G8R8A8 16 = RESERVED16 17 = RESERVED17 18 = RESERVED18 19 = RESERVED19 20 = RESERVED20 21 = RESERVED21 22 = RESERVED22 23 = RESERVED23 24 = RESERVED24 25 = RESERVED25 26 = RESERVED26 27 = RESERVED27 28 = RESERVED28 29 = RESERVED29 30 = RESERVED30 31 = RESERVED31

14.3.24 G2SB_CTX1_G2CONTROLSB_0

Offset: 0x401d │ Byte Offset: 0x10074 │ Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000xxx)

Bit	Reset	Description
31	0x0	DISDW: Output destination writes go either to image memory or the EPP. 0: Output data is sent to memory 1: YUV or RGB data is passed directly to the EPP module, and no destination writes take place. 0 = GOTO_IMAGE_BUFFER 1 = GOTO_EPP
30	0x0	ENDITH: Enable Dithering (ENDITH). For 16 bit RGB output modes, the LSB of the color components can be modified by adding a variable residual value that will reduce the banding artifacts that can appear on the display. 0: Normal operation 1: Enable Dithering 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	0x0	KPOL: Key Signal Polarity (KPOL) Color/Chroma key signal is generated by comparing source input pixel color to a range of a color specified by lower and upper limit values. The key signal is interpreted in two ways, depending on which one of video and graphics images is foreground (and the other is background). This is effective only if Key signal generator is enabled. (see G2CMKEYL, G2CMKEYU) 0: Key signal is set to 1 when source pixel is within the lower and upper limit color range. 1: Key signal is set to 1 when source pixel is outside the lower and upper limit color range. 0 = WITHIN_BOUNDS 1 = OUTSIDE_BOUNDS
27	0x0	KEYEN: Key Signal Generator Enable (KEYEN) Key signal generator generates either chroma key signal (from YCbCr signal) or color key signal (from RGB signal) 0: Key signal generator is disabled. 1: Key signal generator is enabled. 0 = DISABLE 1 = ENABLE
26	0x0	DBSEL: StretchBLT Destination Buffer Selection (DBSEL) StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the destination video area. This bit selects one of the two buffering blocks to which this StretchBLT command delivers the destination image. The two buffering memory blocks are called A and B. 0: Destination image goes to ‘A’ buffer. 1: Destination image goes to ‘B’ buffer. 0 = DST_A 1 = DST_B
25	0x0	SBSEL: StretchBLT Source Buffer Selection (SBSEL) StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the source video area. This bit selects one of the two buffering blocks from which this StretchBLT command receives the source image. The two buffering memory blocks are called source-A and source-B. 0: Source image comes from ‘source-A’ buffer. 1: Source image comes from ‘source-B’ buffer. 0 = SRC_A 1 = SRC_B
24	0x0	SITYPE: StretchBLT Source Type (SITYPE). This bit identifies two types of source images. If source image is 2-to-1 interlaced and StretchBLT processes either one of the two interlaced field-images, physical (positional) displacement between the two interlaced fields must be taken into account. One field-image that is placed higher in position than the other field-image is called top-field and the other is called bottom-field. StretchBLT processing has to lower the top-field (or raise the bottom-field) to match the corresponding two target images in the overlay window (that is progressively scanned) right at the same position. If source image is full frame-image obtained from two interlaced field-images, its type is ‘top-field’. If source images are progressively scanned, the type identification is not significant and they may be designated either one of the two types 0: Source image is ‘top-field’. 1: Source image is ‘bottom-field’. 0 = TOP_FIELD 1 = BOTTOM_FIELD

Bit	Reset	Description
23	0x0	RANGEREDFRM:In the VC-1 specification, when the value of RANGEREDFRM variable (1-bit) for a picture is equal to 1, the picture shall be scaled up according to the following equation:Y = clip(((Y-128)*2) + 128);Cb = clip(((Cb-128)*2) + 128);Cr = clip(((Cr-128)*2) + 128);The clip operator limits the output to [0, 255].The input is also limited to [0, 255].In the VC-1 nomenclature, the output (Y, Cb, Cr) tuple corresponds to the 'decoded' picture. The input (Y, Cb, Cr) tuple corresponds to the 'reconstructed' picture. The above equations create the decoded picture while keeping the reconstructed picture intact.Only YUV _OB formats are supported with range reducation enabled, not the YUV _TC or RGB formats The YUV data coming from the video frame buffers (reconstructed picture) in the SDRAM was scaled up using the above equations and given to the display. the value of 1-bit wide RANGEREDFRM variable was configured in the DVO per frame and the frame was scaled whenever RANGEREDFRM == 1. 0 = DISABLE 1 = ENABLE
22:20	0x0	HFTYPE: StretchBLT Horizontal Filter Mode (HFTYPE[2:0]) The six-tap horizontal interpolation filter can be operated in various operation modes. For the image expansion, it should be programmed as a pure 6-tap interpolator. For the image contraction, it can work as partly low-pass filter and partly interpolater with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of horizontal filtering. 000: Pure interpolation filter 001, 010, 011, 100, 101, 110: mix of interpolation and low pass filters 111: DISABLE. 0 = INTERP 1 = LPF1 2 = LPF2 3 = LPF3 4 = LPF4 5 = LPF5 6 = LPF6 7 = DISABLE
19	0x0	DISCSC: enabled color space converter by default 0 = ENABLE 1 = DISABLE
18	0x0	VFEN: StretchBLT Vertical Filter Enable (VFEN). Vertical filter shall be disabled if source images come from host CPU via CPU Read-FIFO. (SISEL) Vertical filter may be disabled to save some memory bandwidth but this will likely result in degradation of image quality. This option may also be used in the case where source image is progressive scanning and there is no vertical scaling. 0: Vertical filter is disabled. 1: Vertical filter is enabled. 0 = DISABLE 1 = ENABLE
17:16	0x0	VFTYPE: StretchBLT Vertical Filter Mode (VFTYPE[1:0]). This bit is effective when the Vertical Filter is enabled. (bit 18) The two-tap vertical interpolation filter can be operated in various modes. For the image expansion, it should be programmed as a pure 2-tap interpolator. For the image contraction, it can work as partly averager and partly interpolator with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of vertical filtering. 00: Pure interpolation filter. 01: 25% averager, 75% interpolator. 10: 50% averager, 50% interpolator. 11: 100% averager. 0 = INTERP 1 = AVG25_INTERP75 2 = AVG50_INTERP50 3 = AVG

Bit	Reset	Description
9:8	X	UVST: 00: 1/2 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 2 bytes. 01: Equal to Luma Buffer Stride 10: 1/4 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 4 bytes. 11: Use uvstride. Note: Use care when using a tiled surface, since 1/2 or 1/4 of luma stride may yield an invalid pitch (stride). 0 = UVS2X 1 = UVS1X 2 = UVS4X 3 = UVS_G2UVSTRIDE
7	0x0	ENAHF: Enable horizontal alpha filtering if disabled, use the alpha value of third tap for output pixelU,V line stride in 4:2:0 format, 0 = DISABLE 1 = ENABLE
6	0x0	ENAVF: Enable alpha vertical filter 0 = DISABLE 1 = ENABLE
5	0x0	IMODE: Source (input) Data Mode (IMODE). This parameter defines the data mode of source input. 0: Multiplexed Mode (data format is specified by SIFMT[2:0]). One block of source data in the Image Buffer memory contains multiplexed component signals. 1: Planar mode, 4:2:0 YUV (SIFMT[2] specifies data format of U/V components). Three blocks of source data in the Image Buffer memory contain separate Y, U, and V component signals. For a circular buffer input to gr2d, the input format cannot be planar. It must be multiplex, i.e., no YUV420 planar circular buffer input to gr2d. 0 = MULTIPLEX 1 = PLANAR
4	0x0	YUV422PLANAR: Source data is yuv422 planar valid when imode==PLANAR
3	0x0	YUV422ROTATION: YUV422 planar rotated.

14.3.25 G2SB_CTX1_G2CONTROLSECOND_0

Offset: 0x401e │ Byte Offset: 0x10078 │ Read/Write: R/W │ Reset: 0x00000000 (0bxx000000x00xxxx000xxxx000000000x)

Bit	Reset	Description
29	0x0	FR_READWAIT: Fast Rotate wait for reads. Enabling this bit forces FR to wait for the reads to be stored in the data return FIFO before sending writes out to the same block. Always set to ENABLE. This field is debug only. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28:26	0x0	FR_TYPE: Fast Rotate type 000 = FLIP_X 001 = FLIP_Y 010 = TRANS_LR (mirrors about diagonal. Diagonal runs from upper left to lower right) 011 = TRANS_RL (mirrors about diagonal. Diagonal runs from upper right to lower left) 100 = ROT_90 (counterclockwise by 90 degrees) 101 = ROT_180 110 = ROT_270 (clockwise by 90 degrees) 111 = IDENTITY 0 = FLIP_X 1 = FLIP_Y 2 = TRANS_LR 3 = TRANS_RL 4 = ROT_90 5 = ROT_180 6 = ROT_270 7 = IDENTITY
25:24	0x0	FR_MODE: Fast Rotate mode select. G2 Fast Rotate. Transforms a surface via FR_TYPE transformation. Works in either 2-buffer (copy) or 1-buffer mode (in place) mode. The engine breaks down a larger surface into a grid of smaller FR_BLOCKs. Works on the granularity of an FR_BLOCK. An FR_BLOCK is: ▪ 16x16 pixel block (DSTCD = bpp8) ▪ 8x8 pixel block (DSTCD = bpp16) ▪ 4x4 pixel block (DSTCD = bpp32). Maximum surface size is 4096x4096. Key information - source and destination base address must be 128-bit word aligned - engine works on FR_BLOCK granularity: transformed surface width in multiples of 16 bytes** transformed surface height in multiples of 16/8/4 lines for bpp8/bpp16/bpp32 FR_BLOCK if surface dimension is not a multiple, software can program FR engine to transform larger surface (round up to next FR_BLOCK in width and height-->transform-->calculate relative memory pointer address) - during a rotational transformation (TRANS_LR, TRANS_RL, ROT_90, ROT_270): the stride of the output surface != the stride of the input surface when working on non-square input input: output: *^---------- *$---------- $ 0 1 2 3 4 5 6 7 ^ 24 16 8 0 ^ ^ ^ ^ | 8 9 10 11 12 13 14 15 | 25 17 9 1 ^ ^ ^ ^ |16 17 18 19 20 21 22 23 | 26 18 10 2 ^ ^ ^ ^ |24 25 26 27 28 29 30 31 | 27 19 11 3 ^ ^ ^ ^ | - - - - * * * * | 28 20 12 4 * * * * | - - - - * * * * | 29 21 13 5 * * * * | - - - - * * * * | 30 22 14 6 * * * * | - - - - * * * * | 31 23 15 7 * * * * - Tiling alignment restrictions subsume FR restrictions because the FR surface base address is the same as the start address. **Register Programming:** FR_MODE – in place or copy FR_TYPE - type of transformation DSTCD - bpp8, bpp16, bpp32 SRCBA - source base address SRCWIDTH - (width in pixels-1) SRCHEIGHT - (height in lines-1) SRCS - source stride DSTBA - destination base address DSTS - destination stride FR_READWAIT - always set to enable FR inefficency in the following setup: 1. FR_MODE==SQUARE 2. if(FR_TYPE==YFLIP && SRCHEIGHT==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 Or if(FR_TYPE==XFLIP && SRCWIDTH==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 3. Then, the middle 8 lines/pixels, e.g., YFLIP/16bpp/n==3, line 8 to line 15 will be processed twice. 00 = disable fast rotate - this turns off the second level clock to fr_rotate engine. Please remember to do so! 01 = src/dst copy mode - two separate buffers 10 = square in place - one buffer 11 = blank

Bit	Reset	Description
		0 = DISABLE 1 = SRC_DST_COPY 2 = SQUARE 3 = BLANK
22:21	0x0	CLIPC: Clipping rectangle control, if clip enable, bit 57, XYTDW, should be cleared. 0x=clipping disabled, 10=draw only inside clipping rectangle, 11=draw only outside clipping rectangle
9	0x0	ALPSRCORDST: 32-bit blending mode, output alpha selection 0:Source alpha 1:Destination alpha 0 = DISABLE 1 = ENABLE
8:4	0x0	ALPTYPE: Alpha blending method FIX: ALPHA blending: Fixed alpha, ALPHA is the value, B5G6R5 * B5G6R5 VCAA: B5G6R5->B5G6R5 PL1BPP: ALPHA blending: Alpha 1 bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero. PL2BPP: ALPHA blending: Alpha 2 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero. PL4BPP: ALPHA blending: Alpha 4 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero. PL8BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1 bit from source B5G5R5A1, alpha(MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4 bits from source A4B4G4R4, alpha(LSB). Dest: B5G6R5. PLS4BPP: ALPHA blending: Alpha 4 bits from source B4G4R4A4, alpha(MSB). Dest: B5G6R5. VCAA: R8G8B8A8->B5G6R5 without reading VCAA plane (surface blit with bpp down convert - implemented in hardware by the VCAA engine; not really a VCAA resolve) PLS8BPP: ALPHA blending: Alpha 8 bits from source/destination (decided by ALPSRCORDST), R8G8B8A8, alpha(MSB). DST: R8G8B8A8 VCAA: R8G8B8A8->R8G8B8A8, alpha has same blending method as RGB PLS8BX: ALPHA blending: Alpha 8 bits from source B8G8R8A8, alpha(MSB). Dest: B5G6R5. (**Restrictions) VCAA: R8G8B8A8->B5G6R5 PLS1BPPAL: ALPHA blending: Alpha 1 bit from source A1B5G5R5, alpha(LSB). Dest: B5G6R5 VCAA: A1B5G5R5->A1B5G5R5 **Restriction PLS8BX alpha blending has the following restrictions: 1. Source/destination addresses have to be in a 128-bit boundary. 2. Destination width has to be a multiple of 4 pixels. 3. Source/Destination strides have to be a multiple of 128 bits. 0 = FIX 1 = PL1BPP 2 = PL2BPP 3 = PL4BPP 4 = PL8BPP 5 = PL44BPP 6 = PLS1BPP 7 = PLS4BPPAL 8 = PLS4BPP 9 = PLS8BPP 10 = PLS8BX 11 = PLS1BPPAL

Bit	Reset	Description
3	0x0	BEWSWAP: Host port word swap 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 0 = DISABLE 1 = ENABLE

14.3.26 G2SB_CTX1_G2CONTROLMAIN_0

G2 Blit Formats:

(1) G2 Blit size is based on height in lines, width in pixels, stride in bytes, and pixel size (color depth). Unless alpha blending is enabled, G2 does not care about pixel components.

Source color depths supported by G2 Blit:

- Color depth same as destination: G2CONTROLMAIN.SRCCD = 1
- Monochrome: G2CONTROLMAIN.SRCCD = 0

(source monochrome color determined by G2SRCBGC, G2SRCFGC)

Pattern color depths supported by G2 Blit:

- Color depth same as destination: G2PATOS.PATCD = 1
- Monochrome: G2PATOS.PATCD = 0

(pattern monochrome color determined by G2PATBGC, G2PATFGC)

Destination color depths supported by G2 Blit:

- 1 byte per pixel: G2CONTROLMAIN.DSTCD = BPP8
- 2 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP16
- 4 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP32

(2) Alpha blending operates on 3 or 4 component pixels of a specified color depth. One of the components may be alpha (A), depending on the alpha blend format, which is only allowed in certain component positions defined below. The only format rule of the remaining components is that they must be in the same source and destination position because the alpha blend engine does not support component swapping. For example, valid 32BPP formats are: RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888, GRBA_8888->GRBA_8888, etc.

Source formats supported by G2 alpha blend:

- Color depth BPP8: no supported formats
- Color depth BPP16:
  - ALPTYPE_PLS1BPP: xxxA_5551 (for example, BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
  - ALPTYPE_PLS4BPP: xxxA_4444 (for example, BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
  - (other) : xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- Color depth BPP32:
  - (all) : xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

Destination formats supported by G2 alpha blend:

- Color depth BPP8: no supported formats
- Color depth BPP16: xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- Color depth BPP32: xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

src cd	src format	dst cd	dst format	examples
BPP8	no support	BPP8	no support	
BPP16	xyzA_5551, xyzA_4444, xyz_565	BPP16	xyz_565	BGRA_5551->BGR_565, RGB_565->RGB565
BPP32	xyzA_8888	BPP32	xyzA_8888	RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888
BPP32	xyzA_8888	BPP16	xyz_565	RGBA_8888->RGB_565, BGRA_8888->BGR_565

VCAA Resolve Engine

(1) Format support

srcColor format	srcCoverage format	legal dest format(s)
A1B5G5R5	VCAA	A1B5G5R5
R8G8B8A8	VCAA	R8G8B8A8, B5G6R5
B5G6R5	VCAA	B5G6R5

To program resolve type:

- ALPTYPE == FIX: B5G6R5 -> B5G6R5
- ALPTYPE == PLS1BPPAL: A1B5G5R5 -> A1B5G5R5
- ALPTYPE == PLS8BPP: R8G8B8A8 -> R8G8B8A8
- ALPTYPE == PLS8BX: R8G8B8A8 -> B5G6R5

(2) Color surface programming

- color surface base address: SRCBA
- sub-surface start x: SRCX (pixel index)
- sub-surface start y: SRCY (line index)
- color surface stride: SRCS (bytes)

The engine supports subsurface resolve. One can imagine a full color buffer of 1024x768, but

```
[pseudo_code]
      color_surface_start_address_fetch = pcurctx->regs.rG2SRCBA.uSRCBA() +
       (pcurctx->regs.rG2SRCPS.uSRCX() * VCAAState.color_surface_depth) +
       (pcurctx->regs.rG2SRCPS.uSRCY() * VCAAState.color_surface_stride);
```

(3) VCAA surface programming

- VCAA surface base address: PATBA
- VCAA surface stride: PATST (bytes)

```
[pseudo_code]
      vcaa_surface_start_address_fetch = pcurctx->regs.rG2PATBA.uPATBA();
```

Software *MUST* directly program this register to the proper sub-surface location in the vcaa surface which corresponds to the SRCX and SRCY programming. The calculation is PATBA = vcaa_base_address + SRCX * (1) + SRCY * PATST;

(4) Resolve surface programming

- resolve window width: DSTWIDTH (pixels)
- resolve window height: DSTHEIGHT (lines)

- resolve output stride: DSTS (bytes)

These values program the width and height of the resolved surface. In the base case,

- DSTWIDTH = color surface width
- DSTHEIGHT = color sufrace height
- DSTS = resolve surface format bpp * DSTWIDTH

The VCAA engine technically supports DSTX and DSTY (like SRCX and SRCY).

```
[pseudo_code]
   resolve_surface_start_address_put = pcurctx->regs.rG2DSTBA.uDSTBA() +
     (pcurctx->regs.rG2DSTPS.uDSTX() * resolve_surface_color_depth) +
     (pcurctx->regs.rG2DSTPS.uDSTY() * resolve_surface_color_stride);
```

(5) Resolving a pixel

The coverage surface is an 8 bits per pixel surface (C4X4). The surface is initialized to all 1s by 3d.

For a given pixel, we need to look up 4 bits of coverage data. If the data is 0xf, the pixel does not need to be resolved. Otherwise, we need to reblend the pixel with certain weightings of its neighbors (i.e. we're on the edge of some geometry).

If any coverage bit is 0, we need to resolve the color. To do so, calculate the following equation:

```
color_new = (
              20*color_old +
              27*(
                  ((cover_down  == 0) ? color_down  : color_old) +
                  ((cover_right == 0) ? color_right : color_old) +
                  ((cover_left  == 0) ? color_left  : color_old) +
                  ((cover_up    == 0) ? color_up    : color_old)
                 )
            ) / 128;
```

If coverage bit == 0, then we use the neighbor color value in the resolve

If coverage bit == 1, then we use the center color value in the resolve

The resolve is done on a per channel (r/g/b/a) basis:

- format convert input channel to 8-bit
- perform resolve equation above
- format convert to output format

For the resolution of the alpha channel:

- In B5G6R5: No alpha information. Do nothing.
- In A1B5G5R5: No resolve. Keep the source color alpha bit.
- In R8G8B8A8: Normal 8-bit channel resolve

(6) Resolving a pixel on the edge

When on an edge, the vcaa engine treats the coverage bit as a 1. Use the center color value. This holds true even on the edge of the sub-surface resolve window when located in the middle of the larger surface

(7) Maximum dimensions

The maximum resolve sub-surface is 4096 pixels wide.

(8) Surface restrictions

The VCAA engine has the following restrictions with respect to input surfaces

- input base address is 128-bit memory word aligned

- input stride is a multiple of 128 bits (16 bytes)

The VCAA engine has the following restrictions with respect to output surfaces

- output base address is 128-bit memory word aligned

- output stride is a multiple of 128 bits (16 bytes)

(9) Coverage surface layout

The surface is C4X4.

Surface layout (C4X4)

```
(lsb2msb)  C0_C1_C2_C3_X_X_X_X     (X is a don't care.  Qrast inits to 1)
(msb2lsb)  X_X_X_X_C3_C2_C1_C0
    +          o          o
         C0
            C1   Cov(0,1)
       C2
         C3
    o          o          o

      Cov(1,0)   Cov(1,1)

    o          o          o

    +/o = pixel centers
```

In the example above,

```
        Cov(0,0) = X_X_X_X_C3_C2_C1_C0
        C3       = V_C3(Cov(0,0));
        ..         ..
        C0       = V_C0(Cov(0,0));
    #define V_C3(bits) (bits & 0x8)
    #define V_C2(bits) (bits & 0x4)
    #define V_C1(bits) (bits & 0x2)
    #define V_C0(bits) (bits & 0x1)
```

Resolving a pixel

In an ideal world, the coverage would be simply laid out such that when resolving Pixel(x,y), we would need only to fetch Cov(x,y) to get all 4 bits of information.

In actuality, the coverage surface was rearranged to help qrast in performance. To resolve Pixel(x,y) gr2d has to fetch 4 C4X4 bytes [Cov(x,y), Cov(x-1,y), Cov(x,y-1), Cov(x-1,y-1)]. From each C4X4 byte, it extracts one bit which is used in the resolve equations shown in (5).

Need the logical VCAA bits (DRLU) for blend which is spread across four C4X4 coverage bytes

```
          U
    L
        o
            R
      D
```

where

o = Pixel(x,y)

D = V_C1(Cov(x-1,y))

R = V_C0(Cov(x,y))

L = V_C3(Cov(x-1,y-1))

U = V_C2(Cov(x,y-1))

Offset: 0x401f | Byte Offset: 0x1007c | Read/Write: R/W | Reset: 0x00000000 (0bx000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: Pattern Data Select: SRCSEL and PATSEL cannot be both enabled. 0 = SCREEN 1 = MEMORY
28	0x0	DSTDIR: Destination direct addressing
27	0x0	SRCDIR: Source direct addressing
26:25	0x0	GCSW: Display Switching Window Control (GCSW[1:0]). This parameter controls multi-buffering for Display. x0: At end of current command, do not send signal to Display to switch buffer. 01: Two buffers, dstba and dstba_b are used 11: Three buffers, dstba, dstba_b, dstba_c are used
24	X	SRCBAS: Source base address select: 0=srcba 1=dstba. This is not used for Line Draw and if source data comes from host memory.
23:22	0x0	SRCT: Source transparency enable: 0x=Source transparency disabled, 10=Mono source background transparency or color source transparency, 11=Mono source foreground transparency or inverse color source transparency. NOTE: When source transparency is enabled and SRCCD==0(mono) SRCBGC!=SRCFGC should be satisfied.
21	X	HLMONO: Start from Msb or lsb in byte when mono expansion. If HLMONO is 1, bit 0 (the lsb) is the first bit. If HLMONO is 0, bit 7 (the msb) is the first bit
20	X	SRCCD: 0: Source mono 1: Source has same color depth as destination SRCCD==0 xdir/ydir has to be 0
19:18	0x0	DSTT: Destination read transparency enable: 0x=destination read transparency disabled 10=color destination read transparency

Bit	Reset	Description
		11=inverse color destination read transparency. 20 rw INIC Initiate Command (1=initiate command immediately, 0=wait for launch write)
17:16	X	DSTCD: Destination color depth: 00:8-bpp 01:16-bpp 10:32-bpp 11:reserved 0 = BPP8 1 = BPP16 2 = BPP32 3 = RESERVED3
15	0x0	SRCSEL: Source Data Select: 0 = SCREEN 1 = MEMORY
14	0x0	YFLIP: Flip y direction to make image upside down or the other way. If YFLIP==1, ROP cannot include destination. 0 = DISABLE 1 = ENABLE
13	0x0	PATPACK: Pattern Data is in Pack Mode. PATLNGAP in G2PATPACK is the line gap for pattern packed mode. If (PATPACK && ~PATSEl), pattern data is packed and from screen, PATMONOW/PATMONOH should be programmed properly to fetch pattern data from frame buffer. Note, PACK is not supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
12	0x0	SRCPACK: Source Data is in Pack Mode. SRCLNGAP in G2SRCPACK is the line gap for source packed mode. If (SRCPACK && ~SRCSEL), source data is packed and from screen, SRCMONOW/SRCMONOH should be programmed properly to fetch data from frame buffer. Note, PACK is not supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
11	0x0	XYTDW: xy transpose, Line stride DSTS has to be 16 bytes aligned if enabled. If XYTDW==1, ROP cannot include destination.
10	0x0	YDIR: 0=Incrementing 1=Decrementing. ydir should not be set when source surface has different color depth as destination surface. SRCCD==0 (mono src) OR PATCD==0 (mono pat), ydir has to be 0
9	0x0	XDIR: 0=Incrementing 1=Decrementing. xdir should not be set when source surface has different color depth as destination surface. e.g. ALPTYPE=PLS8BX (32bpp blending with 16bpp), xdir has to be 0 SRCCD==0 (mono src) OR PATCD==0 (mono pat), xdir has to be 0
8	0x0	PATFL: When mono pattern is set, we use mono tile pattern fill. Current pattern client can support a 16x16 bit tile which can be used to generate a larger surface (i.e., tile this surface in the x and y direction). The tile is stored in memory. Currently, PATXO and PATYO indicate where to start in the 16x16 tile when expanding the monochrome data. See patxo and patyo comments below for programming When xdir==1 Patxo = pattern width - (patxo+destination width)&0xF -- For xoffsets y offsets PATYO remains the same when xdir/ydir change. How the tile replication pattern looks based on xdir/ydir (without programming patxo above): xdir=0 ydir=0 xdir=1 ydir=0 +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ // xdir=0 ydir=1 xdir=1 ydir=1 Mono tile is +---+ |^|^| |~~~| +---+

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
7	0x0	PATSLD: BitBlt Solid Pattern Fill: 1=enable. BGC will be used as the color value. 0 = DISABLE 1 = ENABLE
6	0x0	SRCSLD: BitBlt Solid Source Color Fill: 1=enable. FGC will be used as the color value. 0 = DISABLE 1 = ENABLE
5	0x0	ALPEN: BitBlt Alpha Blending: 1=enable. 0=disable When both Faden and alpen are 1, output=Source*alpha_v + fadoff.alpha_v is decided by alptype.
4	0x0	FADEN: BitBlt Source Copy Fade enable. 1=enable (share with mltln), only support 16bpp mode 0 = DISABLE 1 = ENABLE
3	0x0	TEST0BIT: Command finish timing bit. 0: 2D command finishes when last data has been pushed to memory write client. 1: 2D command waits memory write client to be idle to finish. 0 = DISABLE 1 = ENABLE
2	0x0	TURBOFILL: Fast fill rectangle in 128 bits/clock. Some limitations with this mode:srcsld==1 rop==0xcc, no clipping, no transparency, xdir==0, ydir==0, flip==0, xytdw==0.Results are undefined if the above limitations are not satisfied.
1:0	0x0	CMDT: Command Type: 00=BitBlt 01=Line Draw 10=VCAA 11=reserved When the raise command is executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

14.3.27 G2SB_CTX1_G2ROPFADE_0

Offset: 0x4020 │ Byte Offset: 0x10080 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:24	X	FADOFF:
23:16	X	FADCOE:
7:0	0x0	ROP: If YFLIP==1 or XYTDW==1, ROP cannot include destination. Since destination may have been corrupted before reading out.

14.3.28 G2SB_CTX1_G2ALPHABLEND_0

Offset: 0x4021 │ Byte Offset: 0x10084 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
31:24	X	ALPHA1V
23:16	X	ALPHA0V
8	0x0	ALPHAINV
7:0	X	ALPHA

14.3.29 G2SB_CTX1_G2CLIPLEFTTOP_0

ClipRect Methods. Top-left bounds are inclusive

Offset: 0x4022 | Byte Offset: 0x10088 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPT
14:0	X	CLIPL

14.3.30 G2SB_CTX1_G2CLIPRIGHTBOT_0

Bottom-right bounds are exclusive

Offset: 0x4023 | Byte Offset: 0x1008c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPB
14:0	X	CLIPR

14.3.31 G2SB_CTX1_G2PATPACK_0

Pattern packed mode

Pattern Methods: G2PATPACK should be used to specify the line gap. Use G2PATPACK_SIZE to program height and width. PACK is generally only useful with narrow monochrome surfaces

> **Note:** PACK is not officially supported when the surface is tiled.

Offset: 0x4024 | Byte Offset: 0x10090 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

14.3.32 G2SB_CTX1_G2PATPACK_SIZE_0

Pattern packed mode

Packed mode, pattern data line gap. Byte

G2PATPACK_SIZE -- extension of G2PATPACK, this register only holds the size of the packed data

> **Note:** PACK is not officially supported when the surface is tiled.

Offset: 0x4025 | Byte Offset: 0x10094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	PATMONOH: Pattern mono data height in packed mode, byte
15:0	X	PATMONOW: Mono data width in packed mode, byte

14.3.33 G2SB_CTX1_G2PATBA_0

Offset: 0x4026 | Byte Offset: 0x10098 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBA: If (PATFL==1){ It has to be 16 bytes aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

14.3.34 G2SB_CTX1_G2PATOS_0

Offset: 0x4027 | Byte Offset: 0x1009c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxx00xxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	PATYO: y offset for mono tile pattern fill. see PATFL
27:24	X	PATXO: x offset for mono tile pattern fill. see PATFL
22:21	0x0	PATT: Mono pattern transparency enable: 0x=pattern transparency disabled 10=mono pattern background transparency or color pattern transparency 11=mono pattern foreground transparency or inverse color pattern transparency. NOTE: When pattern transparency is enabled and PATCD==0(mono) PATBGC!=PATFGC should be satisfied.
16	X	PATCD: 0: mono 1: same as dstcd PATCD==0, xdir/ydir has to be 0
15:0	X	PATST: stride

14.3.35 G2SB_CTX1_G2PATBGC_0

Offset: 0x4028 | Byte Offset: 0x100a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

14.3.36 G2SB_CTX1_G2PATFGC_0

Offset: 0x4029 | Byte Offset: 0x100a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

14.3.37 G2SB_CTX1_G2PATKEY_0

Offset: 0x402a | Byte Offset: 0x100a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

14.3.38 G2SB_CTX1_G2DSTBA_0

Offset: 0x402b | Byte Offset: 0x100ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA: Destination base address (byte address)

14.3.39 G2SB_CTX1_G2DSTBA_B_0

Offset: 0x402c | Byte Offset: 0x100b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_B: Destination base address (byte address), only usable in hardware trigger mode by enable gcsw

14.3.40 G2SB_CTX1_G2DSTBA_C_0

Offset: 0x402d | Byte Offset: 0x100b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_C: Destination base address (byte address)

14.3.41 G2SB_CTX1_G2DSTST_0

Offset: 0x402e | Byte Offset: 0x100b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

14.3.42 G2SB_CTX1_G2SRCPACK_0

Source Data Packed Mode

Surface methods: G2SRCPACK should only be used to specify the line gap. Use G2SRCPACK_SIZE to program height and width

Offset: 0x402f | Byte Offset: 0x100bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

14.3.43 G2SB_CTX1_G2SRCPACK_SIZE_0

Source Data Packed Mode

In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, the required width/stride is 16 byte aligned. Packed mode - source mono data line gap

G2SRCPACKS_SIZE -- extension of G2SRCPACK, this register only holds the size of the packed data

Offset: 0x4030 | Byte Offset: 0x100c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCMONOH: Packed mode - source mono data height in bytes
15:0	X	SRCMONOW: Packed mode - source mono data width in bytes

14.3.44 G2SB_CTX1_G2SRCBA_0

Source Base Address

Offset: 0x4031 | Byte Offset: 0x100c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA: Source base address (byte address)

14.3.45 G2SB_CTX1_G2SRCBA_B_0

SB only

Offset: 0x4032 | Byte Offset: 0x100c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA_B: This parameter specifies the start address of source image stored in the image buffer memory. In 4:2:0 format mode, this image block accommodates for Y-image. This address specifies byte position; however, bits [2:0] are restricted with respect to the data formats to fit multiple pixels in one memory word (8 bytes). For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format. Since one Y pixel takes 8-bit, all 8 byte positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there are no restrictions on this value.)

14.3.46 G2SB_CTX1_G2SRCST_0

Offset: 0x4033 | Byte Offset: 0x100cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS: Source Stride coordinate (bytes) with respect to SRCBA. In order to fit multiple pixels in one memory word (8 bytes), bits [2:0] are restricted with respect to the data formats. For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format.

14.3.47 G2SB_CTX1_G2SRCBGC_0

srcColors

Offset: 0x4034 | Byte Offset: 0x100d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description

31:0	X	SRCBGC

14.3.48 G2SB_CTX1_G2SRCFGC_0

Offset: 0x4035 | Byte Offset: 0x100d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

14.3.49 G2SB_CTX1_G2SRCKEY_0

Offset: 0x4036 | Byte Offset: 0x100d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

14.3.50 G2SB_CTX1_G2SRCSIZE_0

Offset: 0x4037 | Byte Offset: 0x100dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	SRCHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	SRCWIDTH

14.3.51 G2SB_CTX1_G2DSTSIZE_0

Offset: 0x4038 | Byte Offset: 0x100e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	DSTHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	DSTWIDTH: In 2D mode, the largest number can be programmed is 0x7fef

14.3.52 G2SB_CTX1_G2SRCPS_0

ImageBlit Methods

Offset: 0x4039 | Byte Offset: 0x100e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCY
15:0	X	SRCX:SRCX[2:0] are ignored in SRCCD==0 (mono expansion). The first bit of the first byte (bit7 if HLMONO==0, or bit0 if HLMONO==1) always expand to DSTX,DSTY

14.3.53 G2SB_CTX1_G2DSTPS_0

Offset: 0x403a | Byte Offset: 0x100e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	DSTY

Bit	Reset	Description
15:0	X	DSTX: When ALPTYPE is PL1BPP, DSTX[2:0] must be zero. When ALPTYPE is PL2BPP, DSTX[1:0] must be zero When ALPTYPE is PL4BPP, DSTX[0:0] must be zero

14.3.54 G2SB_CTX1_G2CBDES_0

Circular Buffer

Offset: 0x403b │ Byte Offset: 0x100ec │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TOPCLIP: top clipping at the first buffer, (buffer start) refer to TOP_CBLINE in G2CBDES2 0 = DISABLE 1 = ENABLE
30:16	X	CBLINE: vertical line number in one buffer
7:0	X	CBCOUNT: This specifies the number of buffers in circular buffer feature

14.3.55 G2SB_CTX1_G2CBSTRIDE_0

Offset: 0x403c │ Byte Offset: 0x100f0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	X	CBUVSTRIDE: Chroma Buffer Stride default is half of luma 00: 1/2 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 2 bytes. 01: Equal to Luma Buffer Stride 10: 1/4 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 4 bytes. 1x: Reserved 0 = CBS2X 1 = CBS1X 2 = CBS4X
23:0	X	CBSTRIDE: Video Buffer Luma (or RGB) Buffer Stride. This is luma buffer stride (in bytes)

14.3.56 G2SB_CTX1_G2LINESETTING_0

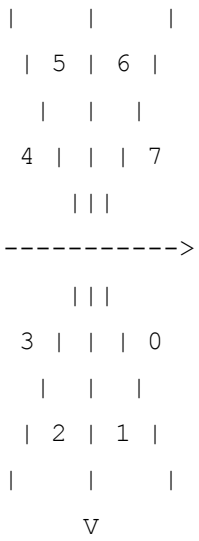
Line Methods

Offset: 0x403d │ Byte Offset: 0x100f4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	OCTANTS: 000: octant 0 001: octant 1 010: octant 2 011: octant 3 100: octant 4 101: octant 5 110: octant 6 111: octant 7

Bit	Reset	Description
28	X	LINEUSEOCTANT: Use OCTANTS in G2LINEDELTAN register instead of MAJOR LINEXDIR LINEYDIR
27	X	DROPLASTP: Draw last pixel or not
26	X	LINEYDIR
25	X	LINEXDIR
24	X	MAJOR: 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

14.3.57 G2SB_CTX1_G2LINEDELTAN_0



Offset: 0x403e | Byte Offset: 0x100f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

14.3.58 G2SB_CTX1_G2LINEDELTAM_0

Offset: 0x403f | Byte Offset: 0x100fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

14.3.59 G2SB_CTX1_G2LINEPOS_0

Offset: 0x4040 | Byte Offset: 0x10100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LINEYPOS
15:0	X	LINEXPOS

14.3.60 G2SB_CTX1_G2LINELEN_0

Offset: 0x4041 | Byte Offset: 0x10104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

14.3.61 G2SB_CTX1_G2CSCFOURTH_0

Offset: 0x4042 | Byte Offset: 0x10108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:16	X	G2V: Multiplier for G for V generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.368 (decimal) or 0x12F. For any other combination, this parameter is ignored.
8:0	X	G2U: Multiplier for G for U generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.291 (decimal) or 0x125. For any other combination, this parameter is ignored.

14.3.62 G2SB_CTX1_G2SRCST_B_0

Offset: 0x4043 | Byte Offset: 0x1010c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

14.3.63 G2SB_CTX1_G2UVSTRIDE_0

Offset: 0x4044 | Byte Offset: 0x10110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

14.3.64 G2SB_CTX1_G2CBDES2_0

Circular Buffer Controller 2

Offset: 0x4045 | Byte Offset: 0x10114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

14.3.65 G2SB_CTX1_G2TILEMODE_0

Offset: 0x4046 | Byte Offset: 0x10118 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx0xxx0xxx0xxx0xxx0xxx0)

Bit	Reset	Description
20	0x0	DST_WR_TILE_MODE: Destination surface 0 = LINEAR 1 = TILED
16	0x0	DST_RD_TILE_MODE: Same as destination write unless DST_RD_WR_SEP (not supported) 0 = LINEAR

Bit	Reset	Description
		1 = TILED
12	0x0	PAT_UV_TILE_MODE:UNUSED 0 = LINEAR 1 = TILED
8	0x0	PAT_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED
4	0x0	SRC_UV_TILE_MODE:UV surface, ignored in RGB mode 0 = LINEAR 1 = TILED
0	0x0	SRC_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED

14.3.66 G2SB_CTX1_G2PATBASE_0

Offset: 0x4047 │ Byte Offset: 0x1011c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PAT_BASE: pattern base address in tile mode, PATBA is the linear address where pixel start

14.3.67 G2SB_CTX1_G2SRCBA_SB_SURFBASE_0

SB_SURFBASE Registers

These registers need only be programmed when using SB. They point to the base of the various source and destination surfaces. Technically they are only needed when tiling is enabled, but there is no harm in always programming them.

Their counterpart registers (_SB_SURFBASE with SB_SURFBASE stripped off) indicate the location of the first pixel to be sourced or written within the surface.

This register exists to mimic the X, Y (SRCX, SRCY) functionality found in the BitBlt engine.

For example, to get at pixel X, Y:

SRCBA = SRCBA_SB_SURFBASE + Y*stride + X*Bpp

Offset: 0x4048 │ Byte Offset: 0x10120 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRC_ADDR: surface address corresponding to G2SRCBA: -base of interleaved sources (RGB, YUV) -base of Y plane. Only used by the StretchBlit Engine

14.3.68 G2SB_CTX1_G2DSTBA_SB_SURFBASE_0

Offset: 0x4049 │ Byte Offset: 0x10124 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_ADDR: surface address corresponding to G2DSTBA. Only used by the StretchBlit Engine

14.3.69 G2SB_CTX1_G2DSTBA_B_SB_SURFBASE_0

Offset: 0x404a | Byte Offset: 0x10128 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_B_ADDR: surface address corresponding to G2DSTBA_B. Only used by the StretchBlit Engine, and G2CONTROLSB.DBSEL() is enabled

14.3.70 G2SB_CTX1_G2VBA_A_SB_SURFBASE_0

Offset: 0x404b | Byte Offset: 0x1012c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	V_ADDR: Surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of V plane. Only used by the StretchBlit Engine

14.3.71 G2SB_CTX1_G2UBA_A_SB_SURFBASE_0

Offset: 0x404c | Byte Offset: 0x10130 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	U_ADDR: Surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of U plane. Only used by the StretchBlit Engine

14.4 G2SB CTX 2 Registers

14.4.1 G2SB_CTX2_INCR_SYNCPT_0

Memory Controller Tiling definitions

Offset: 0x8000 | Byte Offset: 0x20000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	INDX: syncpt index value

14.4.2 G2SB_CTX2_INCR_SYNCPT_CNTRL_0

Offset: 0x8001 | Byte Offset: 0x20004 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

14.4.3 G2SB_CTX2_INCR_SYNCPT_ERROR_0

Offset: 0x8002 │ Byte Offset: 0x20008 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

14.4.4 G2SB_CTX2_G2CLASSCHANNEL_REGONLY_0

This register can be accessed by both register read/write and command FIFO in order not to break exsting code, keep suffix REGONLY.

2D syncpt behavior:

- Each 2D context has its own syncpt client.
- OP_DONE increments when 2D's memory write client is idle, meaning all preceding operations are complete.
- RD_DONE behaves like OP_DONE
- REG_WR_SAFE behaves like IMMEDIATE meaning a syncpt increment returns immediately and does not indicate that a register write will NOT block. The implications are that 2D's host FIFO can fill and block access to 2D, which will indeterminately block a direct CPU access.

Offset: 0x8008 │ Byte Offset: 0x20020 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

14.4.5 G2SB_CTX2_G2TRIGGER_0

Control methods, since they are not programmed often, put them in the beginning

Offset: 0x8009 │ Byte Offset: 0x20024 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

14.4.6 G2SB_CTX2_G2TRIGGER1_0

Offset: 0x800a | Byte Offset: 0x20028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

14.4.7 G2SB_CTX2_G2TRIGGER2_0

Offset: 0x800b | Byte Offset: 0x2002c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

14.4.8 G2SB_CTX2_G2CMDSEL_0

Offset: 0x800c | Byte Offset: 0x20030 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxx0xxxxxxxx00x0x0000000xxxx)

Bit	R/W	Reset	Description
28	RW	0x0	PRIORITY: Indicates priority of the context, note VI-triggered contexts take priority over HIGH 0 = LOW 1 = HIGH
27:25	RW	X	LINKVAL
24	RW	0x0	LINKEN: Test purpose
23:16	RW	X	BUFFER_INDEX
15	RW	0x0	FRAME_END
14	RW	0x0	FRAME_START: host writes this bit to tell which buffer is ready
10	RW	0x0	CLIP_SOURCE_TOP_BOTTOM: Reserved for hardware test. Software should not use this bit. 0 = DISABLE 1 = ENABLE
9:8	RW	0x0	G2OUTPUT: 00: to memory 01: to EPP 0 = MEMORY 1 = EPP 2 = RESERVED2 3 = RESERVED3
7	RW	0x0	CBSBDISABLE
6	RW	0x0	HOSTTRIGGER: host trigger the command, host also need program circular buffer related data
5	RW	0x0	VITRIGGER:VI trigger enable, can be circular buffer or full frame
4	RW	0x0	CBENABLE: Circular buffer feature enable 0 = DISABLE 1 = ENABLE
0	RO	X	SBOR2D: 0 = G2 1 = SB

14.4.9 G2SB_CTX2_G2RAISE_0

Offset: 0x800d | Byte Offset: 0x20034 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE: Push back to read FIFO when all commands in the channel are done.

14.4.10 G2SB_CTX2_G2HOSTSET_0

Offset: 0x800f | Byte Offset: 0x2003c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	HSTFPXL: Specifies the first pixel location (in byte) in the first source data entry. In mono mode pixel should always start in byte boundary
3:0	0x0	HSTLNGAP: Specifies distance (in bytes) from last pixel of a line to the first pixel of next line. In mono mode, the value is the nearest byte number. For example, if the gap is 3 bits, HSTLNGAP should be 1 (byte); if the gap is 9 bits, HSTLNGAP should be 2 (bytes)

14.4.11 G2SB_CTX2_G2HOSTFIFO_0

Offset: 0x8010 | Byte Offset: 0x20040 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

14.4.12 G2SB_CTX2_G2VDDA_0

Offset: 0x8011 | Byte Offset: 0x20044 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VDSTEP: Vertical Scaling DDA. Reset value: xxxx-xxxxh. Vertical DDA Step (Increment) Value (VDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for vertical scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_height-1-VDTINI) / (Actual_destination_height-1). Truncate the rest of the bits to keep the 12-bit fraction. Since we have to meet (Actual_source_height-1)*1.0 >= (Actual_destination_height - 1)*VDSTEP + VDTINI. The 6-bit integer allows a maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half line spacing for up to approximately 2000 target lines. For example, image expansion from 240 lines to 400 lines VDSTEP[17:0] = 19'b00_0000_1001_1001_1010 and image contraction from 240 lines to 150 lines VDSTEP[17:0] = 19'b00_0001_1001_1001_1010.

14.4.13 G2SB_CTX2_G2VDDAINI_0

Offset: 0x8012 | Byte Offset: 0x20048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	VDBINI: Vertical DDA Initial Value for Bottom-Field (VDBINI[7:0]) (see description above)
7:0	X	VDTINI: Vertical Scaling DDA Initial Values. Vertical DDA Initial Value for Top-Field (VDTINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for vertical scaling.

Bit	Reset	Description
		Given a pair of consecutive source image lines representing positions 0.0 and 1.0, a destination line-image will be created at any position in [0.0, 1.0], that is specified by the fraction part of the vertical DDA. The 8-bit initial fraction value specifies the position of the first output (destination) line-image located between the first and the second input (source) line-images, respectively, at positions 0.0 and 1.0. This parameter provides a way to compensate relative displacement of top and bottom fields of a source image. Suppose, 2-to-1 interlaced original video field images #1 and #2 are in the image buffer, to be displayed in an overlay window. Assume that the #1 field image is the top-field. Using VDTINI = 8'hC0 for the top-field image and VDBINI = 8'h40 for the bottom-field image, for example, the two images from #1 and #2 fields get mapped to identically positioned destination images in the overlay window.

14.4.14 G2SB_CTX2_G2HDDA_0

Offset: 0x8013 │ Byte Offset: 0x2004c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HDSTEP: Horizontal Scaling DDA. Reset value: xxxx-xxxxh. Horizontal DDA Step (Increment) Value (HDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for horizontal scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_width-1-HDINI) / (Actual_destination_width-1) Truncate the remaining bits to keep the 12-bit fraction. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half pixel-spacing for up to approximately 2000 destination pixels. For example, image expansion from 720 pixels to 800 pixels HDSTEP[17:0] = 19'b000_0000_1110_0110_0111 and image contraction from 720 pixels to 250 pixels HDSTEP[17:0] = 19'b000_0010_1110_0001_0101.

14.4.15 G2SB_CTX2_G2HDDAINILS_0

Offset: 0x8014 │ Byte Offset: 0x20050 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	HDINI: Horizontal Scaling DDA Initial Value. Horizontal DDA Initial Value (HDINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for horizontal scaling. Given a group of six consecutive source pixels that the two pixels at the center representing positions 0.0 and 1.0, a destination pixel will be created at any position in [0.0, 1.0), that is specified by the fraction part of the horizontal DDA. The 8-bit initial fraction value specifies the position of the first output (destination) pixel located between the first and the second input (source) pixels, respectively at position 0.0 and 1.0.For horizontal scaling this value may normally be set to 0.

14.4.16 G2SB_CTX2_G2CSCFIRST_0

The next 3 registers contain 8 parameters used by the YCbCr (YUV) to RGB color space conversion logic. This logic can also be used for RGB gain/gamma correction in RGB to RGB modes, and for luma gain in YUV to YUV modes.

Two new parameters (G2U, G2V) have been added for RGB to YUV conversion. These are in address 0x38. The equations have the form:

if (yuv2rgb)

(r,g,b) = matrix*(y+sbreg_yos,u,v)

matrix: sbreg_cyx, sbreg_cur, sbreg_cvr,
sbreg_cyx, sbreg_cug, sbreg_cvg,
sbreg_cyx, sbreg_cub, sbreg_cvb,

else if (rgb2rgb)

(r,g,b) = matrix*(r,g+sbreg_yos,b)

matrix:	sbreg_cvr,	0,	sbreg_cur,
	sbreg_cvg,	sbreg_cyx,	sbreg_cug,
	sbreg_cvb,	0,	sbreg_cub,

else if (yuv2yuv)

(y,u,v) = matrix*(y,u,v)

matrix:	sbreg_cub,	0,	sbreg_cvb,
	sbreg_cug,	sbreg_cyx,	sbreg_cvg,
	sbreg_cur,	0,	sbreg_cvr,

else if (rgb2yuv)

(y,u,v) = matrix*(r,g,b) + (sbreg_yos,0,0)

matrix:	sbreg_cvb,	sbreg_cyx,	sbreg_cub,
	sbreg_cvg,	sbreg_g2u,	sbreg_cug,
	sbreg_cvr,	sbreg_g2v,	sbreg_cur,

Offset: 0x8015 │ Byte Offset: 0x20054 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	YOS: Y-Offset (YOFFSET[7:0]) for YUV generation. This parameter consists of 8-bit 2's complement in the range [-128,127]. For YUV->RGB, the recommended value is -16 (decimal) or 0xF0. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is +16 (decimal) or 0x10
21:12	X	CVR: Multiplier for V/R for V or R generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7) For YUV->RGB, the recommended value is 1.5960 (decimal) or 0x0CC If source data is in RGB format, this parameter may be used as gain adjustment for the R component. For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038
9:0	X	CUB: Multiplier for U/B for Y or B generation. Consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 2.0172 (decimal) or 0x102. If source data is in RGB format, this parameter may be used as gain adjustment for the B component. This register changes precision when doing an RGB to YUV conversion (SIFMT=1xxx, DIFMT=0xxx). CUB becomes s1.8 and the recommended value is +0.098 or 0x019

14.4.17 G2SB_CTX2_G2CSCSECOND_0

Offset: 0x8016 │ Byte Offset: 0x20058 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	CYX: Multiplier for Y/G (G gain). This positive-only parameter consists of an 8-bit magnitude (1.7). For YUV->YUV, the recommended value is 1.1644 (decimal) or 0x95. For YUV->YUV or RGB->RGB, this may be used as a gain adjustment for the Y or G component. For RGB->YUV, the recommended value is +0.504 (decimal) or 0x041.
21:12	X	CUR: Multiplier for U/B for V or R generation. This parameter consists of a sign bit and 9-bit

Bit	Reset	Description
		magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This parameter takes a non-zero value if the hue is rotated. For YUV->YUV and RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.071 (decimal) or 0x209
8:0	X	CUG: Multiplier for U/B for U or G generation. Consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB, the recommended value is -0.3918 (decimal) or 0x132. For RGB->RGB, this parameter should be set to 0. For YUV->YUV, this parameter should be set to 1 (0x080). For RGB->YUV the recommended value is +0.439 (decimal) or 0x038

14.4.18 G2SB_CTX2_G2CSCTHIRD_0

Offset: 0x8017 │ Byte Offset: 0x2005c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	CVB: Multiplier for V/R for Y or B generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This coefficient takes a non-zero value if the hue is rotated. For YUV->YUV and RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is +0.257 (decimal) or 0x021.
8:0	X	CVG: Multiplier for V/R for U or G generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB, the recommended value is -0.8130 (decimal) or 0x168. For YUV->YUV and RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.148 (decimal) or 0x113.

14.4.19 G2SB_CTX2_G2CMKEYL_0

When key generation is enabled, the value of this register is the lower color/chroma limit and G2CMKEYU is the upper color/chroma limit for component pixel data. Three component signal values (YCrCb/YUV or RGB) of every output pixel are compared to a set of three ranges that are specified by three pairs of lower and upper color/chroma key values. If the Key Polarity is 0, the Key is set to 1 only when all three component values of an input pixel are respectively within the set of ranges (inclusive of the limit values), else the Key is set to 0. If the Key Polarity is 1, the Key is set to 1 when any of the three component values of an input pixel are outside the set of ranges, else the Key is set to 0.

StretchBLT, Color/Chroma Key Lower Limit Register

Offset: 0x8018 │ Byte Offset: 0x20060 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRL: R or Cr Color Chroma Key Lower Limit (CKRL[7:0]) Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGL: G or Cb Color Chroma Key Lower Limit (CKGL[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
7:0	X	CKBL: B or Y Color/Chroma Key Lower Limit (CKBL[7:0])

14.4.20 G2SB_CTX2_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0x8019 | Byte Offset: 0x20064 | Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRU: R or Cr Color Chroma Key Upper Limit (CKRU[7:0]). The Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGU: G or Cb Color/Chroma Key Upper Limit (CKGU[7:0]). The Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
7:0	X	CKBU: This is the B or Y color/chroma key upper limit value.

14.4.21 G2SB_CTX2_G2UBA_A_0

StretchBLT, Source Image Data U base address A-buffer

Offset: 0x801a | Byte Offset: 0x20068 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SU1SA: Start Address of Source U-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.4.22 G2SB_CTX2_G2VBA_A_0

Offset: 0x801b | Byte Offset: 0x2006c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SV1SA: Start Address of Source V-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since the memory client will assemble YUV into one 4:2:2 format.

14.4.23 G2SB_CTX2_G2SBFORMAT_0

Offset: 0x801c | Byte Offset: 0x20070 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12:8	X	DIFMT: Destination Image Data Format This parameter defines the data format of destination output. There are two groups of data formats: RGB and YCbCr (YUV) formats. CbCr (UV) components may be represented in either offset binary or 2's complement. 00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 01000: bpp16 5-6-5 {R[4:0], G[5:0], B[4:0]} 01001: RESERVED

Bit	Reset	Description
		01010: RESERVED 01011: RESERVED 01100: bpp16 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]} 01101: RESERVED 01110: R8G8B8A801111 B8G8R8A8 1xxxx: RESERVED 0 = U8Y8V8Y8_OB 1 = Y8U8Y8V8_OB 2 = Y8V8Y8U8_OB 3 = V8Y8U8Y8_OB 4 = U8Y8V8Y8_TC 5 = Y8U8Y8V8_TC 6 = Y8V8Y8U8_TC 7 = V8Y8U8Y8_TC 8 = B5G6R5 9 = RESERVED9 10 = RESERVED10 11 = RESERVED11 12 = B5G6R5BS 13 = RESERVED13 14 = R8G8B8A8 15 = B8G8R8A8 16 = RESERVED16 17 = RESERVED17 18 = RESERVED18 19 = RESERVED19 20 = RESERVED20 21 = RESERVED21 22 = RESERVED22 23 = RESERVED23 24 = RESERVED24 25 = RESERVED25 26 = RESERVED26 27 = RESERVED27 28 = RESERVED28 29 = RESERVED29 30 = RESERVED30 31 = RESERVED31
4:0	X	SIFMT: This parameter defines the data format of source input. There are two groups of data formats: RGB format and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement. 00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 01000: B5G6R5 5-6-5 {R[4:0], G[5:0], B[4:0]} 01001: RESERVED 01010: RESERVED 01011: RESERVED 01100: B5G6R5 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]} 01101: RESERVED 01110: R8G8B8A8 01111: B8G8R8A8 1xxxx: RESERVED **StretchBlit Inputs:** RGB inputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV420

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>input = {YUV420 is converted into 4:2:2 UYVY via memory client} YUV422 inputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}

StretchBlit Outputs: RGB outputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV422 outputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}

StretchBlit Input/Output Rules
<table>
<tr><th>src format</th><th>internal sb format</th><th>dst format</th></tr>
<tr><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td><td>R8G8B8A8</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
</table>
StretchBlit color space converter supports YUV->RGB, YUV->YUV (gain) and RGB->RGB (gain). There is no support for RGB->YUV.
0 = U8Y8V8Y8_OB
1 = Y8U8Y8V8_OB
2 = Y8V8Y8U8_OB
3 = V8Y8U8Y8_OB
4 = U8Y8V8Y8_TC
5 = Y8U8Y8V8_TC
6 = Y8V8Y8U8_TC
7 = V8Y8U8Y8_TC
8 = B5G6R5
9 = RESERVED9
10 = RESERVED10
11 = RESERVED11
12 = B5G6R5BS
13 = RESERVED13
14 = R8G8B8A8
15 = B8G8R8A8
16 = RESERVED16
17 = RESERVED17
18 = RESERVED18
19 = RESERVED19
20 = RESERVED20
21 = RESERVED21
22 = RESERVED22
23 = RESERVED23
24 = RESERVED24
25 = RESERVED25
26 = RESERVED26
27 = RESERVED27
28 = RESERVED28
29 = RESERVED29
30 = RESERVED30
31 = RESERVED31</td></tr>
</table>

14.4.24 G2SB_CTX2_G2CONTROLSB_0

Offset: 0x801d │ Byte Offset: 0x20074 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000xxx)

Bit	Reset	Description
31	0x0	DISDW: Output destination writes go either to image memory or the EPP. 0: Output data is sent to memory 1: YUV or RGB data is passed directly to the EPP module, and no destination writes take place. 0 = GOTO_IMAGE_BUFFER 1 = GOTO_EPP
30	0x0	ENDITH: Enable Dithering (ENDITH). For 16-bit RGB output modes, the LSB of the color components can be modified by adding a variable residual value that will reduce the banding artifacts that can appear on the display. 0: Normal operation 1: Enable Dithering 0 = DISABLE 1 = ENABLE
28	0x0	KPOL: Key Signal Polarity (KPOL). Color/Chroma key signal is generated by comparing source input pixel color to a range of a color specified by lower and upper limit values. The key signal is interpreted in two ways, depending on which one of video and graphics images is foreground (and the other is background). This is effective only if Key signal generator is enabled. (see G2CMKEYL, G2CMKEYU) 0: Key signal is set to 1 when source pixel is within the lower and upper limit color range. 1: Key signal is set to 1 when source pixel is outside the lower and upper limit color range. 0 = WITHIN_BOUNDS 1 = OUTSIDE_BOUNDS
27	0x0	KEYEN: Key Signal Generator Enable (KEYEN). Key signal generator generates either chroma key signal (from YCbCr signal) or color key signal (from RGB signal) 0: Key signal generator is disabled. 1: Key signal generator is enabled. 0 = DISABLE 1 = ENABLE
26	0x0	DBSEL: StretchBLT Destination Buffer Selection (DBSEL). StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the destination video area. This bit selects one of the two buffering blocks to which this StretchBLT command delivers the destination image. The two buffering memory blocks are called A and B. 0: Destination image goes to ‘A’ buffer. 1: Destination image goes to ‘B’ buffer. 0 = DST_A 1 = DST_B
25	0x0	SBSEL: StretchBLT Source Buffer Selection (SBSEL). StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the source video area. This bit selects one of the two buffering blocks from which this StretchBLT command receives the source image. The two buffering memory blocks are called source-A and source-B. 0: Source image comes from ‘source-A’ buffer. 1: Source image comes from ‘source-B’ buffer. 0 = SRC_A 1 = SRC_B
24	0x0	SITYPE: StretchBLT Source Type (SITYPE). This bit identifies two types of source images. If source image is 2-to-1 interlaced and StretchBLT processes either one of the two interlaced

Bit	Reset	Description
		field-images, physical (positional) displacement between the two interlaced fields must be taken into account. One field image that is placed higher in position than the other field image is called top field and the other is called bottom field. StretchBLT processing has to lower the top field (or raise the bottom field) to match the corresponding two target images in the overlay window (that is progressively scanned) right at the same position. If source image is a full frame image obtained from two interlaced field-images, its type is 'top-field'. If source images are progressively scanned, the type identification is not significant and they may be designated either one of the two types 0: Source image is 'top field'. 1: Source image is 'bottom field'. 0 = TOP_FIELD 1 = BOTTOM_FIELD
23	0x0	RANGEREDFRM:In the VC-1 specification, when the value of RANGEREDFRM variable (1-bit) for a picture is equal to 1, the picture shall be scaled up according to the following equation:Y = clip(((Y-128)*2) + 128);Cb = clip(((Cb-128)*2) + 128);Cr = clip(((Cr-128)*2) + 128). The clip operator limits the output to [0, 255].The input is also limited to [0, 255]. In the VC-1 nomenclature, the output (Y, Cb, Cr) tuple corresponds to the 'decoded' picture. The input (Y, Cb, Cr) tuple corresponds to the 'reconstructed' picture. The above equations create the decoded picture while keeping the reconstructed picture intact.Only YUV _OB formats are supported with range reducation enabled, not the YUV _TC or RGB formats The YUV data coming from the video frame buffers (reconstructed picture) in the SDRAM was scaled up using the above equations and given to the display. The value of 1-bit wide RANGEREDFRM variable was configured in the DVO per frame and the frame was scaled whenever RANGEREDFRM == 1. 0 = DISABLE 1 = ENABLE
22:20	0x0	HFTYPE: StretchBLT Horizontal Filter Mode (HFTYPE[2:0]). The six-tap horizontal interpolation filter can be operated in various operation modes. For the image expansion, it should be programmed as a pure 6-tap interpolator. For the image contraction, it can work as partly lowpass filter and partly interpolater with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of horizontal filtering. 000: Pure interpolation filter. 001, 010, 011, 100, 101, 110: mix of interpolation and low pass filters 111: DISABLE. 0 = INTERP 1 = LPF1 2 = LPF2 3 = LPF3 4 = LPF4 5 = LPF5 6 = LPF6 7 = DISABLE
19	0x0	DISCSC: Enabled color space converter by default 0 = ENABLE 1 = DISABLE
18	0x0	VFEN: StretchBLT Vertical Filter Enable (VFEN). Vertical filter shall be disabled if source images come from host CPU via CPU Read-FIFO. (SISEL) Vertical filter may be disabled to save some memory bandwidth but this will likely result in degradation of image quality. This option may also be used in the case where source image is progressive scanning and there is no vertical scaling. 0: Vertical filter is disabled. 1: Vertical filter is enabled. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
17:16	0x0	VFTYPE: StretchBLT Vertical Filter Mode (VFTYPE[1:0]). This bit is effective when the Vertical Filter is enabled. (bit 18) The two-tap vertical interpolation filter can be operated in various modes. For the image expansion, it should be programmed as a pure 2-tap interpolator. For the image contraction, it can work as partly averager and partly interpolator with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of vertical filtering. 00: Pure interpolation filter. 01: 25% averager, 75% interpolator. 10: 50% averager, 50% interpolator. 11: 100% averager. 0 = INTERP 1 = AVG25_INTERP75 2 = AVG50_INTERP50 3 = AVG
9:8	X	UVST: 00= 1/2 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 2 bytes. 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 4 bytes. 11= use uvstride. Note: Use care when using a tiled surface, since 1/2 or 1/4 of luma stride may yield an invalid pitch (stride). 0 = UVS2X 1 = UVS1X 2 = UVS4X 3 = UVS_G2UVSTRIDE
7	0x0	ENAHF: Enable horizontal alpha filtering if disabled, use the alpha value of the third tap for output pixelU,V line stride in 4:2:0 Format. 0 = DISABLE 1 = ENABLE
6	0x0	ENAVF:Enable alpha vertical filter 0 = DISABLE 1 = ENABLE
5	0x0	IMODE: Source (input) Data Mode (IMODE). This parameter defines the data mode of source input. 0: Multiplexed Mode (data format is specified by SIFMT[2:0]). One block of source data in the Image Buffer memory contains multiplexed component signals. 1: Planar mode, 4:2:0 YUV (SIFMT[2] specifies data format of U/V components). Three blocks of source data in the Image Buffer memory contain separate Y, U, and V component signals. For a circular buffer input to gr2d, the input format cannot be planar. It must be multiplex, i.e., no YUV420 planar circular buffer input to gr2d. 0 = MULTIPLEX 1 = PLANAR
4	0x0	YUV422PLANAR: Source data is yuv422 planar valid when imode==PLANAR
3	0x0	YUV422ROTATION: YUV422 planar rotated.

14.4.25 G2SB_CTX2_G2CONTROLSECOND_0

Offset: 0x801e │ Byte Offset: 0x20078 │ Read/Write: R/W │ Reset: 0x00000000 (0bxx000000x00xxxx000xxxx000000000x)

Bit	Reset	Description
29	0x0	FR_READWAIT: Fast Rotate wait for reads. Enabling this bit forces FR to wait for the reads to be stored in the data return FIFO before sending writes out to the same block. Always set to ENABLE. This field is debug only. 0 = DISABLE 1 = ENABLE
28:26	0x0	FR_TYPE: Fast Rotate type 000 = FLIP_X 001 = FLIP_Y 010 = TRANS_LR (mirrors about diagonal. Diagonal runs from upper left to lower right) 011 = TRANS_RL (mirrors about diagonal. Diagonal runs from upper right to lower left) 100 = ROT_90 (counter clockwise by 90 degrees) 101 = ROT_180 110 = ROT_270 (clockwise by 90 degrees) 111 = IDENTITY 0 = FLIP_X 1 = FLIP_Y 2 = TRANS_LR 3 = TRANS_RL 4 = ROT_90 5 = ROT_180 6 = ROT_270 7 = IDENTITY
25:24	0x0	FR_MODE: Fast Rotate mode select. G2 Fast Rotate. Transforms a surface via FR_TYPE transformation. Works in either 2-buffer (copy) or 1-buffer mode (in place) mode. The engine breaks down a larger surface into a grid of smaller FR_BLOCKs. Works on the granularity of an FR_BLOCK. An FR_BLOCK is: ▪ 16x16 pixel block (DSTCD = bpp8) ▪ 8x8 pixel block (DSTCD = bpp16) ▪ 4x4 pixel block (DSTCD = bpp32) Maximum surface size is 4096x4096 Key information - source and destination base address must be 128-bit word aligned - engine works on FR_BLOCK granularity: transformed surface width in multiples of 16 bytes** transformed surface height in multiples of 16/8/4 lines for bpp8/bpp16/bpp32 FR_BLOCK if surface dimension is not a multiple, software can program the FR engine to transform larger surface (round up to next FR_BLOCK in width and height-->transform-->calculate relative memory pointer address) - during a rotational transformation (TRANS_LR, TRANS_RL, ROT_90, ROT_270): the stride of the output surface != the stride of the input surface when working on non-square input input: output: *^---------- *$---------- $ 0 1 2 3 4 5 6 7 ^ 24 16 8 0 ^ ^ ^ ^ | 8 9 10 11 12 13 14 15 | 25 17 9 1 ^ ^ ^ ^ |16 17 18 19 20 21 22 23 | 26 18 10 2 ^ ^ ^ ^ |24 25 26 27 28 29 30 31 | 27 19 11 3 ^ ^ ^ ^ | - - - - * * * * | 28 20 12 4 * * * * | - - - - * * * * | 29 21 13 5 * * * * | - - - - * * * * | 30 22 14 6 * * * * | - - - - * * * * | 31 23 15 7 * * * * - Tiling alignment restrictions subsume FR restrictions because the FR surface base address is the same as the start address. **Register Programming**: FR_MODE - inplace or copy FR_TYPE - type of transformation DSTCD - bpp8, bpp16, bpp32 SRCBA - source base address SRCWIDTH - (width in pixels-1) SRCHEIGHT - (height in lines-1) SRCS - source stride DSTBA - destination base address DSTS - destination stride FR_READWAIT - always set to enable FR inefficiency in the following setup :

Bit	Reset	Description
		1. FR_MODE==SQUARE 2. if(FR_TYPE==YFLIP && SRCHEIGHT==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 Or if(FR_TYPE==XFLIP && SRCWIDTH==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 3. Then, the middle 8 lines/pixels, e.g., YFLIP/16bpp/n==3, line 8 to line 15 will be processed twice. 00 = Disable fast rotate - this turns off the second level clock to fr_rotate engine. Please remember to do so! 01 = src/dst copy mode - two separate buffers 10 = square in place - one buffer 11 = blank 0 = DISABLE 1 = SRC_DST_COPY 2 = SQUARE 3 = BLANK
22:21	0x0	CLIPC: Clipping rectangle control, if clip enable, bit 57, XYTDW, should be cleared. 0x=clipping disabled 10=draw only inside clipping rectangle 11=draw only outside clipping rectangle
9	0x0	ALPSRCORDST: 32-bit blending mode, output alpha selection 0:Source alpha 1:Destination alpha 0 = DISABLE 1 = ENABLE
8:4	0x0	ALPTYPE: Alpha blending method FIX: ALPHA blending: Fixed alpha, ALPHA is the value, B5G6R5 * B5G6R5 VCAA: B5G6R5->B5G6R5 PL1BPP: ALPHA blending: Alpha 1 bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero. PL2BPP: ALPHA blending: Alpha 2 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero. PL4BPP: ALPHA blending: Alpha 4 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero. PL8BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1 bit from source B5G5R5A1, alpha (MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4 bits from source A4B4G4R4, alpha (LSB). Dest: B5G6R5. PLS4BPP: ALPHA blending: Alpha 4bits from source B4G4R4A4, alpha (MSB). Dest: B5G6R5. VCAA: R8G8B8A8->B5G6R5 without reading VCAA plane (surface blit with bpp down convert - implemented in hardware by the VCAA engine; not really a VCAA resolve) PLS8BPP: ALPHA blending: Alpha 8 bits from source/destination (decided by ALPSRCORDST), R8G8B8A8, alpha (MSB). DST: R8G8B8A8 VCAA: R8G8B8A8->R8G8B8A8, alpha has same blending method as RGB PLS8BX: ALPHA blending: Alpha 8 bits from source B8G8R8A8, alpha (MSB). Dest: B5G6R5. (**Restrictions) VCAA: R8G8B8A8->B5G6R5 PLS1BPPAL: ALPHA blending: Alpha 1 bit from source A1B5G5R5, alpha (LSB). Dest: B5G6R5 VCAA: A1B5G5R5->A1B5G5R5 **Restriction PLS8BX alpha blending has the following restrictions: 1. Source/destination addresses have to be in a 128-bit boundary. 2. Destination width has to be a multiple of 4 pixels. 3. Source/Destination strides have to be a multiple of 128 bits. 0 = FIX 1 = PL1BPP 2 = PL2BPP

Bit	Reset	Description
		3 = PL4BPP 4 = PL8BPP 5 = PL44BPP 6 = PLS1BPP 7 = PLS4BPPAL 8 = PLS4BPP 9 = PLS8BPP 10 = PLS8BX 11 = PLS1BPPAL
3	0x0	BEWSWAP: Host port word swap 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 0 = DISABLE 1 = ENABLE

14.4.26 G2SB_CTX2_G2CONTROLMAIN_0

G2 Blit Formats:

(1) G2 Blit size is based on height in lines, width in pixels, stride in bytes, and pixel size (color depth). Unless alpha blending is enabled, G2 does not care about pixel components.

Source color depths supported by G2 Blit:

- color depth same as destination: G2CONTROLMAIN.SRCCD = 1
- monochrome: G2CONTROLMAIN.SRCCD = 0

(source monochrome color determined by G2SRCBGC, G2SRCFGC)

Pattern color depths supported by G2 Blit:

- color depth same as destination: G2PATOS.PATCD = 1
- monochrome: G2PATOS.PATCD = 0

(pattern monochrome color determined by G2PATBGC, G2PATFGC)

Destination color depths supported by G2 Blit:

- 1 byte per pixel: G2CONTROLMAIN.DSTCD = BPP8
- 2 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP16
- 4 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP32

(2) Alpha blending operates on 3 or 4 component pixels of a specified color depth. One of the components may be alpha (A), depending on the alpha blend format, which is only allowed in certain component positions defined below. The only format rule of the remaining components is that they must be in the same source and destination position because the alpha blend engine does not support component swapping. For example, valid 32BPP formats are: RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888, GRBA_8888->GRBA_8888, etc.

Source formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16:
  - ALPTYPE_PLS1BPP: xxxA_5551 (for example, BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])

- ALPTYPE_PLS4BPP: xxxA_4444 (for example, BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
- (other) : xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])

- color depth BPP32:
  - (all) : xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

Destination formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16: xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

src cd	src format	dst cd	dst format	examples
BPP8	no support	BPP8	no support	
BPP16	xyzA_5551, xyzA_4444, xyz_565	BPP16	xyz_565	BGRA_5551->BGR_565, RGB_565->RGB565
BPP32	xyzA_8888	BPP32	xyzA_8888	RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888
BPP32	xyzA_8888	BPP16	xyz_565	RGBA_8888->RGB_565, BGRA_8888->BGR_565

VCAA Resolve Engine

(1) Format support

srcColor format	srcCoverage format	legal dest format(s)
A1B5G5R5	VCAA	A1B5G5R5
R8G8B8A8	VCAA	R8G8B8A8, B5G6R5
B5G6R5	VCAA	B5G6R5

To program resolve type:

- ALPTYPE == FIX: B5G6R5 -> B5G6R5
- ALPTYPE == PLS1BPPAL: A1B5G5R5 -> A1B5G5R5
- ALPTYPE == PLS8BPP: R8G8B8A8 -> R8G8B8A8
- ALPTYPE == PLS8BX: R8G8B8A8 -> B5G6R5

(2) Color surface programming

- color surface base address: SRCBA
- subsurface start x: SRCX (pixel index)
- subsurface start y: SRCY (line index)
- color surface stride: SRCS (bytes)

The engine supports subsurface resolve. One can imagine a full color buffer of 1024x768, but

```
[pseudo_code]
      color_surface_start_address_fetch = pcurctx->regs.rG2SRCBA.uSRCBA() +
       (pcurctx->regs.rG2SRCPS.uSRCX() * VCAAState.color_surface_depth) +
       (pcurctx->regs.rG2SRCPS.uSRCY() * VCAAState.color_surface_stride);
```

(3) VCAA surface programming

- VCAA surface base address: PATBA
- VCAA surface stride: PATST (bytes)

```
[pseudo_code]
      vcaa_surface_start_address_fetch = pcurctx->regs.rG2PATBA.uPATBA();
```

Software *MUST* directly program this register to the proper subsurface location in the vcaa surface which corresponds to the SRCX and SRCY programming. The calculation is PATBA = vcaa_base_address + SRCX * (1) + SRCY * PATST;

(4) Resolve surface programming

- resolve window width: DSTWIDTH (pixels)
- resolve window height: DSTHEIGHT (lines)
- resolve output stride: DSTS (bytes)

These values program the width and height of the resolved surface. In the base case,

- DSTWIDTH = color surface width
- DSTHEIGHT = color sufrace height
- DSTS = resolve surface format bpp * DSTWIDTH

The VCAA engine technically supports DSTX and DSTY (like SRCX and SRCY).

```
[pseudo_code]
   resolve_surface_start_address_put = pcurctx->regs.rG2DSTBA.uDSTBA() +
     (pcurctx->regs.rG2DSTPS.uDSTX() * resolve_surface_color_depth) +
     (pcurctx->regs.rG2DSTPS.uDSTY() * resolve_surface_color_stride);
```

(5) Resolving a pixel

The coverage surface is an 8 bits per pixel surface (C4X4). The surface is initialized to all 1s by 3d.

For a given pixel, we need to look up 4 bits of coverage data. If the data is 0xf, the pixel does not need to be resolved. Otherwise, we need to reblend the pixel with certain weightings of its neighbors (i.e., we are on the edge of some geometry).

If any coverage bit is 0, we need to resolve the color. To do so, calculate the following equation:

```
color_new = (
              20*color_old +
              27*(
                  ((cover_down  == 0) ? color_down  : color_old) +
                  ((cover_right == 0) ? color_right : color_old) +
                  ((cover_left  == 0) ? color_left  : color_old) +
                  ((cover_up    == 0) ? color_up    : color_old)
                  )
            ) / 128;
```

If coverage bit == 0, then we use the neighbor color value in the resolve

If coverage bit == 1, then we use the center color value in the resolve

The resolve is done on a per channel (r/g/b/a) basis:

- format convert input channel to 8-bit
- perform resolve equation above
- format convert to output format

For the resolution of the alpha channel:

- In B5G6R5: no alpha information. do nothing.
- In A1B5G5R5: no resolve. keep the source color alpha bit.
- In R8G8B8A8: normal 8-bit channel resolve

(6) Resolving a pixel on the edge

When on an edge, the VCAA engine treats the coverage bit as a 1. Use the center color value. This holds true even on the edge of the sub-surface resolve window when located in the middle of the larger surface

(7) Maximum dimensions

The maximum resolve subsurface is 4096 pixels wide.

(8) Surface restrictions

The VCAA engine has the following restrictions with respect to input surfaces

- input base address is 128-bit memory word aligned
- input stride is a multiple of 128 bits (16 bytes)

The VCAA engine has the following restrictions with respect to output surfaces

- output base address is 128-bit memory word aligned
- output stride is a multiple of 128 bits (16 bytes)

(9) Coverage surface layout

The surface is C4X4.

Surface layout (C4X4)

```
(lsb2msb)  C0_C1_C2_C3_X_X_X_X     (X is a don't care.  Qrast inits to 1)
(msb2lsb)  X_X_X_X_C3_C2_C1_C0


    +          o          o
         C0
            C1   Cov(0,1)
       C2
         C3
    o          o          o

      Cov(1,0)   Cov(1,1)


    o          o          o
    +/o = pixel centers
```

In the example above,

```
       Cov(0,0) = X_X_X_X_C3_C2_C1_C0
       C3       = V_C3(Cov(0,0));
       ..         ..
       C0       = V_C0(Cov(0,0));
    #define V_C3(bits) (bits & 0x8)
    #define V_C2(bits) (bits & 0x4)
    #define V_C1(bits) (bits & 0x2)
```

```
#define V_C0(bits) (bits & 0x1)
```

Resolving a pixel

In an ideal world, the coverage would be simply laid out such that when resolving Pixel(x,y), we would need only to fetch Cov(x,y) to get all 4 bits of information.

In actuality, the coverage surface was rearranged to help qrast in performance. To resolve Pixel(x,y) gr2d has to fetch 4 C4X4 bytes [Cov(x,y), Cov(x-1,y), Cov(x,y-1), Cov(x-1,y-1)]. From each C4X4 byte, it extracts one bit which is used in the resolve equations shown in (5).

Need the logical VCAA bits (DRLU) for blend which is spread across four C4X4 coverage bytes:

```
        U

  L

     o

        R

    D
```

Where:

o = Pixel(x,y)

D = V_C1(Cov(x-1,y))

R = V_C0(Cov(x,y))

L = V_C3(Cov(x-1,y-1))

U = V_C2(Cov(x,y-1))

Offset: 0x801f | Byte Offset: 0x2007c | Read/Write: R/W | Reset: 0x00000000 (0bx000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: Pattern Data Select: SRCSEL and PATSEL cannot be both enabled. 0 = SCREEN 1 = MEMORY
28	0x0	DSTDIR: destination direct addressing
27	0x0	SRCDIR: source direct addressing
26:25	0x0	GCSW: Display Switching Window Control (GCSW[1:0]). This parameter controls multi-buffering for Display. x0: At end of current command, do not send signal to Display to switch buffer. 01: two buffers, dstba and dstba_b are used 11: three buffers, dstba, dstba_b, dstba_c are used
24	X	SRCBAS: Source base address select: 0=srcba 1=dstba. This is not used for Line Draw and if source data comes from host memory.
23:22	0x0	SRCT: Source transparency enable: 0x=source transparency disabled, 10=mono source background transparency or color source transparency, 11=mono source foreground transparency or inverse color source transparency. NOTE: When source transparency is enabled and SRCCD==0(mono) SRCBGC!=SRCFGC should be satisfied.

Bit	Reset	Description
21	X	HLMONO: Start from msb or lsb in byte when mono expansion If HLMONO is 1, bit 0 (the lsb) is the first bit. If HLMONO is 0, bit 7 (the msb) is the first bit
20	X	SRCCD: 0: Source mono 1: Source has same color depth as destination SRCCD==0 xdir/ydir has to be 0
19:18	0x0	DSTT: Destination read transparency enable: 0x=destination read transparency disabled 10=color destination read transparency 11=inverse color destination read transparency. 20 rw INIC Initiate Command (1=initiate command immediately, 0=wait for launch write)
17:16	X	DSTCD: Destination color depth: 00=8-bpp, 01=16-bpp, 10=32-bpp. 11=reserved 0 = BPP8 1 = BPP16 2 = BPP32 3 = RESERVED3
15	0x0	SRCSEL: Source Data Select: 0 = SCREEN 1 = MEMORY
14	0x0	YFLIP: flip y direction to make image upside down or the other way. If YFLIP==1, ROP cannot include destination. 0 = DISABLE 1 = ENABLE
13	0x0	PATPACK: Pattern Data is in Pack Mode. PATLNGAP in G2PATPACK is the line gap for pattern packed mode. If (PATPACK && ~PATSEl), pattern data is packed and from screen, PATMONOW/PATMONOH should be programmed properly to fetch pattern data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
12	0x0	SRCPACK: Source Data is in Pack Mode. SRCLNGAP in G2SRCPACK is the line gap for source packed mode. If (SRCPACK && ~SRCSEL), source data is packed and from screen, SRCMONOW/SRCMONOH should be programmed properly to fetch data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
11	0x0	XYTDW: xy transpose, Line stride DSTS has to be 16-byte aligned if enabled. If XYTDW==1, ROP cannot include destination.
10	0x0	YDIR: 0=incrementing, 1=decrementing. ydir should not be set when source surface has different color depth as destination surface. SRCCD==0 (mono src) OR PATCD==0 (mono pat), ydir has to be 0
9	0x0	XDIR: 0=incrementing, 1=decrementing. xdir should not be set when source surface has different color depth as destination surface. e.g. ALPTYPE=PLS8BX (32bpp blending with 16bpp), xdir has to be 0 SRCCD==0 (mono src) OR PATCD==0 (mono pat), xdir has to be 0

Bit	Reset	Description

8	0x0	PATFL: When mono pattern is set, we use mono tile pattern fill. current pattern client can support a 16x16 bit tile which can be used to generate a larger surface (i.e. tile this surface in the x and y direction) The tile is stored in memory. Currently, PATXO and PATYO indicate where to start in the 16x16 tile when expanding the monochrome data. See patxo and patyo comments below for programming
		When xdir==1 Patxo = pattern width - (patxo+destination width)&0xF -- For xoffsets y offsets PATYO remains the same when xdir/ydir change. How the tile replication pattern looks based on xdir/ydir (without programming patxo above): xdir=0 ydir=0 xdir=1 ydir=0 +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ // xdir=0 ydir=1 xdir=1 ydir=1 Mono tile is +---+ |^|^| |~~~| +---+
		0 = DISABLE
		1 = ENABLE

7	0x0	PATSLD: BitBlt Solid Pattern Fill: 1=enable. BGC will be used as the color value. 0 = DISABLE 1 = ENABLE
6	0x0	SRCSLD: BitBlt Solid Source Color Fill: 1=enable. FGC will be used as the color value. 0 = DISABLE 1 = ENABLE
5	0x0	ALPEN: BitBlt Alpha Blending, 1=enable. 0=disable, when both Faden and alpen are 1, output=Source*alpha_v + fadoff, alpha_v is decided by alptype
4	0x0	FADEN: BitBlt Source Copy Fade enable, 1=enable (share with mltln), only support 16bpp mode 0 = DISABLE 1 = ENABLE
3	0x0	TEST0BIT: Command finish timing bit 0: 2D command finishes when last data has been pushed to memory write client. 1: 2D command waits memory write client to be idle to finish. 0 = DISABLE 1 = ENABLE
2	0x0	TURBOFILL: fast fill rectangle in 128 bits/clock. Some limitations with this mode:srcsld==1 rop==0xcc, no clipping, no transparency, xdir==0, ydir==0, flip==0, xytdw==0. Results are undefined if the above limitations are not satisfied.
1:0	0x0	CMDT: Command Type: 00=BitBlt 01=Line Draw 10=VCAA 11=reserved When the raise command is executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

14.4.27 G2SB_CTX2_G2ROPFADE_0

Offset: 0x8020 │ Byte Offset: 0x20080 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:24	X	FADOFF:
23:16	X	FADCOE:

Bit	Reset	Description
7:0	0x0	ROP: If YFLIP==1 or XYTDW==1, ROP cannot include destination. Since destination may have been corrupted before reading out.

14.4.28 G2SB_CTX2_G2ALPHABLEND_0

Offset: 0x8021 | Byte Offset: 0x20084 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
31:24	X	ALPHA1V
23:16	X	ALPHA0V
8	0x0	ALPHAINV
7:0	X	ALPHA

14.4.29 G2SB_CTX2_G2CLIPLEFTTOP_0

ClipRect Methods. Top-left bounds are inclusive

Offset: 0x8022 | Byte Offset: 0x20088 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPT
14:0	X	CLIPL

14.4.30 G2SB_CTX2_G2CLIPRIGHTBOT_0

Bottom-right bounds are exclusive

Offset: 0x8023 | Byte Offset: 0x2008c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPB
14:0	X	CLIPR

14.4.31 G2SB_CTX2_G2PATPACK_0

Pattern packed mode

Pattern methods: G2PATPACK should be used to specify the line gap

use G2PATPACK_SIZE to program height and width

PACK is generally only useful with narrow monochrome surfaces

Note: PACK is not officially supported when the surface is tiled.

Offset: 0x8024 | Byte Offset: 0x20090 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

14.4.32 G2SB_CTX2_G2PATPACK_SIZE_0

Pattern packed mode

Packed mode, pattern data line gap. Byte

G2PATPACK_SIZE -- extension of G2PATPACK, this register only holds the size of the packed data

Note: PACK is not officially supported when the surface is tiled.

Offset: 0x8025 | Byte Offset: 0x20094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	PATMONOH: Pattern mono data height in packed mode, byte
15:0	X	PATMONOW: Mono data width in packed mode, byte

14.4.33 G2SB_CTX2_G2PATBA_0

Offset: 0x8026 | Byte Offset: 0x20098 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBA: If (PATFL==1){ It has to be 16-byte aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

14.4.34 G2SB_CTX2_G2PATOS_0

Offset: 0x8027 | Byte Offset: 0x2009c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00xxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	PATYO: y offset for mono tile pattern fill. see PATFL
27:24	X	PATXO: x offset for mono tile pattern fill. see PATFL
22:21	0x0	PATT: Mono pattern transparency enable: 0x=pattern transparency disabled, 10=mono pattern background transparency or color pattern transparency, 11=mono pattern foreground transparency or inverse color pattern transparency. NOTE: When pattern transparency is enabled and PATCD==0(mono) PATBGC!=PATFGC should be satisfied.
16	X	PATCD:0 mono1 same as dstcd PATCD==0 xdir/ydir has to be 0
15:0	X	PATST: stride

14.4.35 G2SB_CTX2_G2PATBGC_0

Offset: 0x8028 | Byte Offset: 0x200a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

14.4.36 G2SB_CTX2_G2PATFGC_0

Offset: 0x8029 | Byte Offset: 0x200a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

14.4.37 G2SB_CTX2_G2PATKEY_0

Offset: 0x802a | Byte Offset: 0x200a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

14.4.38 G2SB_CTX2_G2DSTBA_0

Offset: 0x802b | Byte Offset: 0x200ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA: Destination base address (byte address)

14.4.39 G2SB_CTX2_G2DSTBA_B_0

Offset: 0x802c | Byte Offset: 0x200b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_B: Destination base address (byte address) only usable in hardware trigger mode by enable gcsw

14.4.40 G2SB_CTX2_G2DSTBA_C_0

Offset: 0x802d | Byte Offset: 0x200b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_C: Destination base address (byte address)

14.4.41 G2SB_CTX2_G2DSTST_0

Offset: 0x802e | Byte Offset: 0x200b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

14.4.42 G2SB_CTX2_G2SRCPACK_0

Source Data Packed Mode

Surface methods G2SRCPACK should only be used to specify the line gap. Use G2SRCPACK_SIZE to program height and width.

Offset: 0x802f │ Byte Offset: 0x200bc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

14.4.43 G2SB_CTX2_G2SRCPACK_SIZE_0

Source Data Packed Mode

In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, the required width/stride is 16 byte aligned. Packed mode - source mono data line gap

G2SRCPACKS_SIZE -- extension of G2SRCPACK, this register only holds the size of the packed data

Offset: 0x8030 │ Byte Offset: 0x200c0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCMONOH: Packed mode - source mono data height in bytes
15:0	X	SRCMONOW: Packed mode - source mono data width in bytes

14.4.44 G2SB_CTX2_G2SRCBA_0

Source Base Address

Offset: 0x8031 │ Byte Offset: 0x200c4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA: Source base address (byte address)

14.4.45 G2SB_CTX2_G2SRCBA_B_0

SB only

Offset: 0x8032 │ Byte Offset: 0x200c8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA_B: This parameter specifies the start address of source image stored in the image buffer memory. In 4:2:0 format mode, this image block accommodates for Y-image. This address specifies byte position; however, bits [2:0] are restricted with respect to the data formats to fit multiple pixels in one memory word (8 bytes). For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format. Since one Y pixel takes 8 bits, all 8 byte positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there are no restrictions on this value.)

14.4.46 G2SB_CTX2_G2SRCST_0

Offset: 0x8033 | Byte Offset: 0x200cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS: Source Stride coordinate (bytes) with respect to SRCBA. In order to fit multiple pixels in one memory word (8 bytes), bits [2:0] are restricted with respect to the data formats. For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format.

14.4.47 G2SB_CTX2_G2SRCBGC_0

srcColors

Offset: 0x8034 | Byte Offset: 0x200d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

14.4.48 G2SB_CTX2_G2SRCFGC_0

Offset: 0x8035 | Byte Offset: 0x200d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

14.4.49 G2SB_CTX2_G2SRCKEY_0

Offset: 0x8036 | Byte Offset: 0x200d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

14.4.50 G2SB_CTX2_G2SRCSIZE_0

Offset: 0x8037 | Byte Offset: 0x200dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	SRCHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	SRCWIDTH

14.4.51 G2SB_CTX2_G2DSTSIZE_0

Offset: 0x8038 | Byte Offset: 0x200e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	DSTHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	DSTWIDTH: In 2D mode, the largest number can be programmed is 0x7fef

14.4.52 G2SB_CTX2_G2SRCPS_0

ImageBlit Methods

Offset: 0x8039 | Byte Offset: 0x200e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCY
15:0	X	SRCX:SRCX[2:0] are ignored in SRCCD==0 (mono expansion). The first bit of the first byte (bit7 if HLMONO==0, or bit0 if HLMONO==1) always expand to DSTX,DSTY

14.4.53 G2SB_CTX2_G2DSTPS_0

Offset: 0x803a | Byte Offset: 0x200e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	DSTY
15:0	X	DSTX: NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero

14.4.54 G2SB_CTX2_G2CBDES_0

Circular Buffer

Offset: 0x803b | Byte Offset: 0x200ec | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TOPCLIP: top clipping at the first buffer, (buffer start) refer to TOP_CBLINE in G2CBDES2 0 = DISABLE 1 = ENABLE
30:16	X	CBLINE: vertical line number in one buffer
7:0	X	CBCOUNT: This specifies the number of buffers in circular buffer feature

14.4.55 G2SB_CTX2_G2CBSTRIDE_0

Offset: 0x803c | Byte Offset: 0x200f0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	X	CBUVSTRIDE: Chroma Buffer Stride default is half of luma 00= 1/2 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 2 bytes. 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 4 bytes. 1x= Reserved 0 = CBS2X 1 = CBS1X 2 = CBS4X
23:0	X	CBSTRIDE: Video Buffer Luma(or RGB) Buffer Stride. This is luma buffer stride (in bytes)

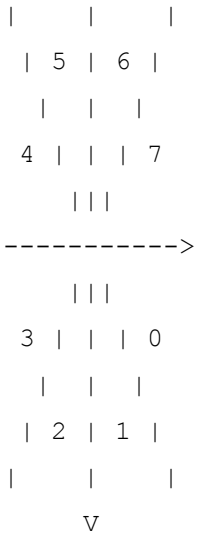
14.4.56 G2SB_CTX2_G2LINESETTING_0

Line Methods

Offset: 0x803d | Byte Offset: 0x200f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	OCTANTS: 000: octant 0 001: octant 1 010: octant 2 011: octant 3 100: octant 4 101: octant 5 110: octant 6 111: octant 7
28	X	LINEUSEOCTANT: Use OCTANTS in G2LINEDELTAN register instead of MAJOR LINEXDIR LINEYDIR
27	X	DROPLASTP: Draw last pixel or not
26	X	LINEYDIR
25	X	LINEXDIR
24	X	MAJOR: 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

14.4.57 G2SB_CTX2_G2LINEDELTAN_0



Offset: 0x803e | Byte Offset: 0x200f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

14.4.58 G2SB_CTX2_G2LINEDELTAM_0

Offset: 0x803f | Byte Offset: 0x200fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

14.4.59 G2SB_CTX2_G2LINEPOS_0

Offset: 0x8040 | Byte Offset: 0x20100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LINEYPOS
15:0	X	LINEXPOS

14.4.60 G2SB_CTX2_G2LINELEN_0

Offset: 0x8041 | Byte Offset: 0x20104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

14.4.61 G2SB_CTX2_G2CSCFOURTH_0

Offset: 0x8042 | Byte Offset: 0x20108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:16	X	G2V: Multiplier for G for V generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.368 (decimal) or 0x12F. For any other combination, this parameter is ignored.
8:0	X	G2U: Multiplier for G for U generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.291 (decimal) or 0x125. For any other combination, this parameter is ignored.

14.4.62 G2SB_CTX2_G2SRCST_B_0

Offset: 0x8043 | Byte Offset: 0x2010c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

14.4.63 G2SB_CTX2_G2UVSTRIDE_0

Offset: 0x8044 | Byte Offset: 0x20110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

14.4.64 G2SB_CTX2_G2CBDES2_0

Circular Buffer Controller 2

Offset: 0x8045 | Byte Offset: 0x20114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

14.4.65 G2SB_CTX2_G2TILEMODE_0

Offset: 0x8046 | Byte Offset: 0x20118 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx0xxx0xxx0xxx0xxx0xxx0)

Bit	Reset	Description
20	0x0	DST_WR_TILE_MODE: destination surface 0 = LINEAR 1 = TILED
16	0x0	DST_RD_TILE_MODE: Same as destination write unless DST_RD_WR_SEP (not supported) 0 = LINEAR 1 = TILED
12	0x0	PAT_UV_TILE_MODE:UNUSED 0 = LINEAR 1 = TILED
8	0x0	PAT_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED
4	0x0	SRC_UV_TILE_MODE:UV surface, ignored in RGB mode 0 = LINEAR 1 = TILED
0	0x0	SRC_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED

14.4.66 G2SB_CTX2_G2PATBASE_0

Offset: 0x8047 | Byte Offset: 0x2011c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PAT_BASE: Pattern base address in tile mode, PATBA is the linear address where pixel start

14.4.67 G2SB_CTX2_G2SRCBA_SB_SURFBASE_0

SB_SURFBASE Registers

These registers need only be programmed when using SB. They point to the base of the various source and destination surfaces. Technically they are only needed when tiling is enabled, but there is no harm in always programming them.

Their counterpart registers (_SB_SURFBASE with SB_SURFBASE stripped off) indicate the location of the first pixel to be sourced or written within the surface.

This register exists to mimic the X, Y (SRCX, SRCY) functionality found in the BitBlt engine.

For example, to get at pixel X, Y:

SRCBA = SRCBA_SB_SURFBASE + Y*stride + X*Bpp

Offset: 0x8048 | Byte Offset: 0x20120 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRC_ADDR: Surface address corresponding to G2SRCBA: -base of interleaved sources (RGB, YUV) -base of Y plane. Only used by the StretchBlit Engine

14.4.68 G2SB_CTX2_G2DSTBA_SB_SURFBASE_0

Offset: 0x8049 | Byte Offset: 0x20124 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_ADDR: Surface address corresponding to G2DSTBA. Only used by the StretchBlit Engine

14.4.69 G2SB_CTX2_G2DSTBA_B_SB_SURFBASE_0

Offset: 0x804a | Byte Offset: 0x20128 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_B_ADDR: Surface address corresponding to G2DSTBA_B. Only used by the StretchBlit Engine, and G2CONTROLSB.DBSEL() is enabled

14.4.70 G2SB_CTX2_G2VBA_A_SB_SURFBASE_0

Offset: 0x804b | Byte Offset: 0x2012c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	V_ADDR: Surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of V plane. Only used by the StretchBlit Engine

14.4.71 G2SB_CTX2_G2UBA_A_SB_SURFBASE_0

Offset: 0x804c | Byte Offset: 0x20130 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	U_ADDR: Surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of U plane. Only used by the StretchBlit Engine

14.5 G2SB CTX 3 Registers

14.5.1 G2SB_CTX3_INCR_SYNCPT_0

Memory Controller Tiling definitions

Offset: 0x1000 | Byte Offset: 0x4000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	INDX: syncpt index value

14.5.2 G2SB_CTX3_INCR_SYNCPT_CNTRL_0

Offset: 0x1001 | Byte Offset: 0x4004 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

14.5.3 G2SB_CTX3_INCR_SYNCPT_ERROR_0

Offset: 0x1002 | Byte Offset: 0x4008 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

14.5.4 G2SB_CTX3_G2CLASSCHANNEL_REGONLY_0

This register can be accessed by both register read/write and command FIFO in order not to break exsting code , keep suffix REGONLY.

2D syncpt behavior

- Each 2D context has its own syncpt client.
- OP_DONE increments when 2D's memory write client is idle, meaning all preceding operations are complete.

- RD_DONE behaves like OP_DONE
- REG_WR_SAFE behaves like IMMEDIATE meaning a syncpt increment returns immediately and does not indicate that a register write will NOT block. The implications are that 2D's host FIFO can fill and block access to 2D, which will indeterminately block a direct CPU access.

Offset: 0x1008 | Byte Offset: 0x4020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

14.5.5 G2SB_CTX3_G2TRIGGER_0

Control methods, since they are not programmed often, put them is the beginning

Offset: 0x1009 | Byte Offset: 0x4024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

14.5.6 G2SB_CTX3_G2TRIGGER1_0

Offset: 0x100a | Byte Offset: 0x4028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

14.5.7 G2SB_CTX3_G2TRIGGER2_0

Offset: 0x100b | Byte Offset: 0x402c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

14.5.8 G2SB_CTX3_G2CMDSEL_0

Offset: 0x100c | Byte Offset: 0x4030 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxx0xxxxxxxx00x0x0000000xxxx)

Bit	R/W	Reset	Description
28	RW	0x0	PRIORITY: indicates priority of the context, note vi-triggered contexts take priority over HIGH 0 = LOW 1 = HIGH
27:25	RW	X	LINKVAL
24	RW	0x0	LINKEN: Test purpose
23:16	RW	X	BUFFER_INDEX
15	RW	0x0	FRAME_END
14	RW	0x0	FRAME_START: host writes this bit to tell which buffer is ready

Bit	R/W	Reset	Description
10	RW	0x0	CLIP_SOURCE_TOP_BOTTOM: Reserved for hardware test. Software should not use this bit. 0 = DISABLE 1 = ENABLE
9:8	RW	0x0	G2OUTPUT: 00: to memory 01: to EPP 0 = MEMORY 1 = EPP 2 = RESERVED2 3 = RESERVED3
7	RW	0x0	CBSBDISABLE
6	RW	0x0	HOSTTRIGGER: host trigger the command, host also need program circular buffer related data
5	RW	0x0	VITRIGGER:VI trigger enable, can be circular buffer or full frame
4	RW	0x0	CBENABLE: circular buffer feature enable 0= disable 0 = DISABLE 1 = ENABLE
0	RO	X	SBOR2D: 0 = G2 1 = SB

14.5.9 G2SB_CTX3_G2RAISE_0

Offset: 0x100d │ Byte Offset: 0x4034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE: Push back to read FIFO when all commands in the channel are done.

14.5.10 G2SB_CTX3_G2HOSTSET_0

Offset: 0x100f │ Byte Offset: 0x403c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	HSTFPXL: specifies the first pixel location (in byte) in the first source data entry. In mono mode pixel should always start in byte boundary
3:0	0x0	HSTLNGAP: specifies distance (in byte) from last pixel of a line to the first pixel of next line. In mono mode, the value is the nearest byte number. e.g., if gap is 3 bits, HSTLNGAP should be 1 (byte), if gap is 9 bits, HSTLNGAP should be 2 (byte)

14.5.11 G2SB_CTX3_G2HOSTFIFO_0

Offset: 0x1010 │ Byte Offset: 0x4040 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

14.5.12 G2SB_CTX3_G2VDDA_0

Offset: 0x1011 | Byte Offset: 0x4044 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VDSTEP: Vertical Scaling DDA. Reset value: xxxx-xxxxh. Vertical DDA Step (Increment) Value (VDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for vertical scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_height-1-VDTINI) / (Actual_destination_height-1). Truncate the remaining bits to keep the 12-bit fraction. Since we have to meet (Actual_source_height-1)*1.0 >= (Actual_destination_height - 1)*VDSTEP + VDTIN. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half line-spacing for up to approximately 2000 target lines. For example, image expansion from 240 lines to 400 lines VDSTEP[17:0] = 19'b00_0000_1001_1001_1010 and image contraction from 240 lines to 150 lines VDSTEP[17:0] = 19'b00_0001_1001_1001_1010.

14.5.13 G2SB_CTX3_G2VDDAINI_0

Offset: 0x1012 | Byte Offset: 0x4048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	VDBINI: Vertical DDA Initial Value for Bottom-Field (VDBINI[7:0]) (see description above)
7:0	X	VDTINI: Vertical Scaling DDA Initial Values. Vertical DDA Initial Value for Top-Field (VDTINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for vertical scaling. Given a pair of consecutive source image lines representing positions 0.0 and 1.0, a destination line-image will be created at any position in [0.0, 1.0), that is specified by the fraction part of the vertical DDA. The 8-bit initial fraction value specifies the position of the first output (destination) line-image located between the first and the second input (source) line-images, respectively at position 0.0 and 1.0. This parameter provides a way to compensate relative displacement of top and bottom fields of a source image. Suppose, 2-to-1 interlaced original video field images #1 and #2 are in the image buffer, to be displayed in an overlay window. Assume that the #1 field image is the top-field. Using VDTINI = 8'hC0 for the top-field image and VDBINI = 8'h40 for the bottom-field image, for example, the two images from #1 and #2 fields get mapped to identically positioned destination images in the overlay window.

14.5.14 G2SB_CTX3_G2HDDA_0

Offset: 0x1013 | Byte Offset: 0x404c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HDSTEP: Horizontal Scaling DDA. Reset value: xxxx-xxxxh. Horizontal DDA Step (Increment) Value (HDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for horizontal scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_width-1-HDINI) / (Actual_destination_width-1). Truncate the remaining bits to keep the 12-bit fraction. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half pixel-spacing for up to approximately 2000 destination pixels. For example, image expansion from 720 pixels to 800 pixels HDSTEP[17:0] = 19'b000_0000_1110_0110_0111 and image contraction from 720 pixels to 250 pixels HDSTEP[17:0] = 19'b000_0010_1110_0001_0101.

14.5.15 G2SB_CTX3_G2HDDAINILS_0

Offset: 0x1014 | Byte Offset: 0x4050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	HDINI: Horizontal Scaling DDA Initial Value. Horizontal DDA Initial Value (HDINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for horizontal scaling. Given a group of six consecutive source pixels that the two pixels at the center representing positions 0.0 and 1.0, a destination pixel will be created at any position in [0.0, 1.0), that is specified by the fraction part of the horizontal DDA. The 8-bit initial fraction value specifies the position of the first output (destination) pixel located between the first and the second input (source) pixels, respectively at position 0.0 and 1.0.For horizontal scaling this value may normally be set to 0.

14.5.16 G2SB_CTX3_G2CSCFIRST_0

The next 3 registers contain 8 parameters used by the YCbCr (YUV) to RGB color space conversion logic. This logic can also be used for RGB gain/gamma correction in RGB to RGB modes, and for luma gain in YUV to YUV modes.

Two new parameters (G2U, G2V) have been added for RGB to YUV conversion. These are in address 0x38. The equations have the form:

if (yuv2rgb)

(r,g,b) = matrix*(y+sbreg_yos,u,v)

matrix: sbreg_cyx, sbreg_cur, sbreg_cvr,
sbreg_cyx, sbreg_cug, sbreg_cvg,
sbreg_cyx, sbreg_cub, sbreg_cvb,

else if (rgb2rgb)

(r,g,b) = matrix*(r,g+sbreg_yos,b)

matrix: sbreg_cvr, 0, sbreg_cur,
sbreg_cvg, sbreg_cyx, sbreg_cug,
sbreg_cvb, 0, sbreg_cub,

else if (yuv2yuv)

(y,u,v) = matrix*(y,u,v)

matrix: sbreg_cub, 0, sbreg_cvb,
sbreg_cug, sbreg_cyx, sbreg_cvg,
sbreg_cur, 0, sbreg_cvr,

else if (rgb2yuv)

(y,u,v) = matrix*(r,g,b) + (sbreg_yos,0,0)

matrix: sbreg_cvb, sbreg_cyx, sbreg_cub,
sbreg_cvg, sbreg_g2u, sbreg_cug,
sbreg_cvr, sbreg_g2v, sbreg_cur,

Offset: 0x1015 | Byte Offset: 0x4054 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	YOS: Y-Offset (YOFFSET[7:0]) for YUV generation This parameter consists of 8-bit 2's complement in the range [-128,127]. For YUV->RGB, the recommended value is -16 (decimal) or 0xF0. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is +16 (decimal) or 0x10

Bit	Reset	Description
21:12	X	CVR: Multiplier for V/R for V or R generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 1.5960 (decimal) or 0x0CCIf source data is in RGB format, this parameter may be used as gain adjustment for R component. For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038
9:0	X	CUB: Multiplier for U/B for Y or B generation. Consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 2.0172 (decimal) or 0x102. If source data is in RGB format, this parameter may be used as gain adjustment for B component. This register changes precision when doing RGB to YUV conversion (SIFMT=1xxx, DIFMT=0xxx). CUB becomes s1.8 and the recommended value is +0.098 or 0x019

14.5.17 G2SB_CTX3_G2CSCSECOND_0

Offset: 0x1016 │ Byte Offset: 0x4058 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	CYX: Multiplier for Y/G (G gain). This positive-only parameter consists of 8-bit magnitude (1.7). For YUV->YUV the recommended value is 1.1644 (decimal) or 0x95. For YUV->YUV or RGB->RGB, this may be used as gain adjustment for Y or G component. For RGB->YUV the recommended value is +0.504 (decimal) or 0x041
21:12	X	CUR: Multiplier for U/B for V or R generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This parameter takes non-zero value if hue is rotated. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.071 (decimal) or 0x209.
8:0	X	CUG: Multiplier for U/B for U or G generation. Consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB, the recommended value is -0.3918 (decimal) or 0x132. For RGB->RGB, this parameter should be set to 0. For YUV->YUV, this parameter should be set to 1 (0x080). For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038.

14.5.18 G2SB_CTX3_G2CSCTHIRD_0

Offset: 0x1017 │ Byte Offset: 0x405c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	CVB: Multiplier for V/R for Y or B generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This coefficient takes non-zero value if hue is rotated. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is +0.257 (decimal) or 0x021
8:0	X	CVG: Multiplier for V/R for U or G generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB, the recommended value is -0.8130 (decimal) or 0x168. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.148 (decimal) or 0x113

14.5.19 G2SB_CTX3_G2CMKEYL_0

When key generation is enabled, the value of this register is the lower color/chroma limit and G2CMKEYU is the upper color/chroma limit for component pixel data. Three component signal values (YCrCb/YUV or RGB) of every output pixel are compared to a set of three ranges that are specified by three pairs of lower and upper color/chroma key values. If the Key Polarity is 0, the Key is set to 1 only when all three component values of an input pixel are respectively within the set of ranges (inclusive of the limit values), else the Key is set to 0. If the Key Polarity is 1, the Key is set to 1 when any of the three component values of an input pixel are outside the set of ranges, else the Key is set to 0.

StretchBLT, Color/Chroma Key Lower Limit Register

Offset: 0x1018 | Byte Offset: 0x4060 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRL: R or Cr Color Chroma Key Lower Limit (CKRL[7:0]) Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGL: G or Cb Color Chroma Key Lower Limit (CKGL[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
7:0	X	CKBL: B or Y Color/Chroma Key Lower Limit (CKBL[7:0])

14.5.20 G2SB_CTX3_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0x1019 | Byte Offset: 0x4064 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRU: R or Cr Color Chroma Key Upper Limit (CKRU[7:0]) Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGU: G or Cb Color/Chroma Key Upper Limit (CKGU[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity the minimum signal level to the maximum signal level.
7:0	X	CKBU: This is B or Y color/chroma key upper limit value.

14.5.21 G2SB_CTX3_G2UBA_A_0

StretchBLT, Source Image Data U base address A-buffer

Offset: 0x101a | Byte Offset: 0x4068 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SU1SA: Start Address of Source U-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.5.22 G2SB_CTX3_G2VBA_A_0

Offset: 0x101b | Byte Offset: 0x406c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SV1SA: Start Address of Source V-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.5.23 G2SB_CTX3_G2SBFORMAT_0

Offset: 0x101c │ Byte Offset: 0x4070 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12:8	X	DIFMT: Destination Image Data Format This parameter defines the data format of destination output. There are two groups of data formats, RGB and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement. 00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 01000: bpp16 5-6-5 {R[4:0], G[5:0], B[4:0]} 01001: RESERVED 01010: RESERVED 01011: RESERVED 01100: bpp16 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]} 01101: RESERVED 01110: R8G8B8A8 01111: B8G8R8A8 1xxxx: RESERVED 0 = U8Y8V8Y8_OB 1 = Y8U8Y8V8_OB 2 = Y8V8Y8U8_OB 3 = V8Y8U8Y8_OB 4 = U8Y8V8Y8_TC 5 = Y8U8Y8V8_TC 6 = Y8V8Y8U8_TC 7 = V8Y8U8Y8_TC 8 = B5G6R5 9 = RESERVED9 10 = RESERVED10 11 = RESERVED11 12 = B5G6R5BS 13 = RESERVED13 14 = R8G8B8A8 15 = B8G8R8A8 16 = RESERVED16 17 = RESERVED17 18 = RESERVED18 19 = RESERVED19 20 = RESERVED20 21 = RESERVED21 22 = RESERVED22 23 = RESERVED23 24 = RESERVED24 25 = RESERVED25 26 = RESERVED26 27 = RESERVED27 28 = RESERVED28 29 = RESERVED29 30 = RESERVED30 31 = RESERVED31

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td>4:0</td><td>X</td><td>
SIFMT: This parameter defines the data format of source input. There are two groups of data formats, RGB format and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement.
00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
01000: B5G6R5 5-6-5 {R[4:0], G[5:0], B[4:0]}
01001: RESERVED
01010: RESERVED
01011: RESERVED
01100: B5G6R5 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]}
01101: RESERVED
01110: R8G8B8A8
01111: B8G8R8A8
1xxxx: RESERVED
StretchBlit Inputs: RGB inputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV420 input = {YUV420 is converted into 4:2:2 UYVY via memory client} YUV422 inputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}
StretchBlit Outputs: RGB outputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV422 outputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}
StretchBlit Input/Output Rules
<table>
<tr><th>src format</th><th>internal sb format</th><th>dst format</th></tr>
<tr><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td><td>R8G8B8A8</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
</table>
StretchBlit color space converter supports YUV->RGB, YUV->YUV (gain) and RGB->RGB (gain). There is no support for RGB->YUV.
0 = U8Y8V8Y8_OB
1 = Y8U8Y8V8_OB
2 = Y8V8Y8U8_OB
3 = V8Y8U8Y8_OB
4 = U8Y8V8Y8_TC
5 = Y8U8Y8V8_TC
6 = Y8V8Y8U8_TC
7 = V8Y8U8Y8_TC
8 = B5G6R5
9 = RESERVED9
10 = RESERVED10
11 = RESERVED11
12 = B5G6R5BS
13 = RESERVED13
14 = R8G8B8A8
15 = B8G8R8A8
16 = RESERVED16
17 = RESERVED17
</td></tr>
</table>

Bit	Reset	Description
		18 = RESERVED18 19 = RESERVED19 20 = RESERVED20 21 = RESERVED21 22 = RESERVED22 23 = RESERVED23 24 = RESERVED24 25 = RESERVED25 26 = RESERVED26 27 = RESERVED27 28 = RESERVED28 29 = RESERVED29 30 = RESERVED30 31 = RESERVED31

14.5.24 G2SB_CTX3_G2CONTROLSB_0

Offset: 0x101d │ Byte Offset: 0x4074 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000xxx)

Bit	Reset	Description
31	0x0	DISDW: Output destination writes go either to image memory or the EPP. 0: Output data is sent to memory 1: YUV or RGB data is passed directly to the EPP module, and no destination writes take place. 0 = GOTO_IMAGE_BUFFER 1 = GOTO_EPP
30	0x0	ENDITH: Enable Dithering (ENDITH). For 16 bit RGB output modes, the LSB of the color components can be modified by adding a variable residual value that will reduce the banding artifacts that can appear on the display. 0 Normal operation 1 Enable Dithering 0 = DISABLE 1 = ENABLE
28	0x0	KPOL: Key Signal Polarity (KPOL). Color/Chroma key signal is generated by comparing source input pixel color to a range of a color specified by lower and upper limit values. The key signal is interpreted in two ways, depending on which one of video and graphics images is foreground (and the other is background). This is effective only if Key signal generator is enabled. (see G2CMKEYL, G2CMKEYU) 0: Key signal is set to 1 when source pixel is within the lower and upper limit color range. 1: Key signal is set to 1 when source pixel is outside the lower and upper limit color range. 0 = WITHIN_BOUNDS 1 = OUTSIDE_BOUNDS
27	0x0	KEYEN: Key Signal Generator Enable (KEYEN) Key signal generator generates either chroma key signal (from YCbCr signal) or color key signal (from RGB signal) 0: Key signal generator is disabled. 1: Key signal generator is enabled. 0 = DISABLE 1 = ENABLE
26	0x0	DBSEL: StretchBLT Destination Buffer Selection (DBSEL) StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the destination video area. This bit selects one of the two buffering blocks to which this StretchBLT command delivers the destination image. The two buffering memory blocks are called A and B. 0: Destination image goes to ‘A’ buffer. 1: Destination image goes to ‘B’ buffer. 0 = DST_A 1 = DST_B

Bit	Reset	Description
25	0x0	SBSEL: StretchBLT Source Buffer Selection (SBSEL) StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the source video area. This bit selects one of the two buffering blocks from which this StretchBLT command receives the source image. The two buffering memory blocks are called source-A and source-B. 0: Source image comes from 'source-A' buffer. 1: Source image comes from 'source-B' buffer. 0 = SRC_A 1 = SRC_B
24	0x0	SITYPE: StretchBLT Source Type (SITYPE). This bit identifies two types of source images. If source image is 2-to-1 interlaced and StretchBLT processes either one of the two interlaced field-images, physical (positional) displacement between the two interlaced fields must be taken into account. One field-image that is placed higher in position than the other field-image is called top-field and the other is called bottom-field. StretchBLT processing has to lower the top-field (or raise the bottom-field) to match the corresponding two target images in the overlay window (that is progressively scanned) right at the same position. If source image is full frame-image obtained from two interlaced field-images, its type is 'top-field'. If source images are progressively scanned, the type identification is not significant and they may be designated either one of the two types 0: Source image is 'top-field'. 1: Source image is 'bottom-field'. 0 = TOP_FIELD 1 = BOTTOM_FIELD
23	0x0	RANGEREDFRM:In the VC-1 specification, when the value of RANGEREDFRM variable (1-bit) for a picture is equal to 1, the picture shall be scaled up according to the following equation:Y = clip(((Y-128)*2) + 128);Cb = clip(((Cb-128)*2) + 128);Cr = clip(((Cr-128)*2) + 128);The clip operator limits the output to [0, 255].The input is also limited to [0, 255]. In the VC-1 nomenclature, the output (Y, Cb, Cr) tuple corresponds to the 'decoded' picture. The input (Y, Cb, Cr) tuple corresponds to the 'reconstructed' picture. The above equations create the decoded picture while keeping the reconstructed picture intact.Only YUV _OB formats are supported with range reducation enabled, not the YUV _TC or RGB formats. The YUV data coming from the video frame buffers (reconstructed picture) in the SDRAM was scaled up using the above equations and given to the display. the value of 1-bit wide RANGEREDFRM variable was configured in the DVO per frame and the frame was scaled whenever RANGEREDFRM == 1. 0 = DISABLE 1 = ENABLE
22:20	0x0	HFTYPE: StretchBLT Horizontal Filter Mode (HFTYPE[2:0]) The six-tap horizontal interpolation filter can be operated in various operation modes. For the image expansion, it should be programmed as a pure 6-tap interpolator. For the image contraction, it can work as partly lowpass filter and partly interpolater with varying degree depending the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of horizontal filtering. 000: Pure interpolation filter. 001, 010, 011, 100, 101, 110: mix of interpolation and low pass filters 111: DISABLE. 0 = INTERP 1 = LPF1 2 = LPF2 3 = LPF3 4 = LPF4 5 = LPF5 6 = LPF6 7 = DISABLE

Bit	Reset	Description
19	0x0	DISCSC: enabled color space converter by default 0 = ENABLE 1 = DISABLE
18	0x0	VFEN: StretchBLT Vertical Filter Enable (VFEN) Vertical filter shall be disabled if source images come from host CPU via CPU Read-FIFO. (SISEL) Vertical filter may be disabled to save some memory bandwidth but this will likely result in degradation of image quality. This option may also be used in the case where source image is progressive scanning and there is no vertical scaling. 0: Vertical filter is disabled. 1: Vertical filter is enabled. 0 = DISABLE 1 = ENABLE
17:16	0x0	VFTYPE: StretchBLT Vertical Filter Mode (VFTYPE[1:0]). This bit is effective when the Vertical Filter is enabled. (bit 18) The two-tap vertical interpolation filter can be operated in various modes. For the image expansion, it should be programmed as a pure 2-tap interpolator. For the image contraction, it can work as partly averager and partly interpolator with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of vertical filtering. 00: Pure interpolation filter 01: 25% averager, 75% interpolator 10: 50% averager, 50% interpolator 11: 100% averager. 0 = INTERP 1 = AVG25_INTERP75 2 = AVG50_INTERP50 3 = AVG
9:8	X	UVST: 00= 1/2 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 2 bytes 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 4 bytes 11= use uvstride. Note: Use care when using a tiled surface, since 1/2 or 1/4 of luma stride may yield an invalid pitch (stride). 0 = UVS2X 1 = UVS1X 2 = UVS4X 3 = UVS_G2UVSTRIDE
7	0x0	ENAHF: enable horizontal alpha filtering if disabled, use the alpha value of third tap for output pixelU,V line stride in 4:2:0 Format, 0 = DISABLE 1 = ENABLE
6	0x0	ENAVF:enable alpha vertical filter 0 = DISABLE 1 = ENABLE
5	0x0	IMODE: Source (input) Data Mode (IMODE). This parameter defines the data mode of source input. 0: Multiplexed Mode (data format is specified by SIFMT[2:0]). One block of source data in the Image Buffer memory contains multiplexed component signals. 1: Planar mode, 4:2:0 YUV (SIFMT[2] specifies data format of U/V components). Three blocks of source data in the Image Buffer memory contain separate Y, U, and V component signals. For a circular buffer input to gr2d, the input format cannot be planar. It must be multiplex, i.e., no YUV420 planar circular buffer input to gr2d. 0 = MULTIPLEX 1 = PLANAR

Bit	Reset	Description
4	0x0	YUV422PLANAR: Source data is yuv422 planar valid when imode==PLANAR
3	0x0	YUV422ROTATION: YUV422 planar rotated.

14.5.25 G2SB_CTX3_G2CONTROLSECOND_0

Offset: 0x101e │ Byte Offset: 0x4078 │ Read/Write: R/W │ Reset: 0x00000000 (0bxx000000x00xxxx000xxxx000000000x)

Bit	Reset	Description
29	0x0	FR_READWAIT: Fast Rotate wait for reads. Enabling this bit forces FR to wait for the reads to be stored in the data return FIFO before sending writes out to the same block. Always set to ENABLE. This field is debug only. 0 = DISABLE 1 = ENABLE
28:26	0x0	FR_TYPE: Fast Rotate type 000 = FLIP_X 001 = FLIP_Y 010 = TRANS_LR (mirrors about diagonal. Diagonal runs from upper left to lower right) 011 = TRANS_RL (mirrors about diagonal. Diagonal runs from upper right to lower left) 100 = ROT_90 (counterclockwise by 90 degrees) 101 = ROT_180 110 = ROT_270 (clockwise by 90 degrees) 111 = IDENTITY 0 = FLIP_X 1 = FLIP_Y 2 = TRANS_LR 3 = TRANS_RL 4 = ROT_90 5 = ROT_180 6 = ROT_270 7 = IDENTITY
25:24	0x0	FR_MODE: Fast Rotate mode select. G2 Fast Rotate. Transforms a surface via FR_TYPE transformation. Works in either 2-buffer (copy) or 1-buffer mode (in place) mode. The engine breaks down a larger surface into a grid of smaller FR_BLOCKs. Works on the granularity of an FR_BLOCK. An FR_BLOCK is: ▪ 16x16 pixel block (DSTCD = bpp8) ▪ 8x8 pixel block (DSTCD = bpp16) ▪ 4x4 pixel block (DSTCD = bpp32) Maximum surface size is 4096x4096 Key information - source and destination base address must be 128-bit word aligned - engine works on FR_BLOCK granularity: transformed surface width in multiples of 16 bytes** transformed surface height in multiples of 16/8/4 lines for bpp8/bpp16/bpp32 FR_BLOCK if surface dimension is not a multiple, software can program the FR engine to transform larger surface (round up to next FR_BLOCK in width and height-->transform-->calculate relative memory pointer address) - during a rotational transformation (TRANS_LR, TRANS_RL, ROT_90, ROT_270): the stride of the output surface != the stride of the input surface when working on non-square input input: output: *^---------- *$---------- $ 0 1 2 3 4 5 6 7 ^ 24 16 8 0 ^ ^ ^ ^ | 8 9 10 11 12 13 14 15 | 25 17 9 1 ^ ^ ^ ^ |16 17 18 19 20 21 22 23 | 26 18 10 2 ^ ^ ^ ^ |24 25 26 27 28 29 30 31 | 27 19 11 3 ^ ^ ^ ^ | - - - - * * * * | 28 20 12 4 * * * * | - - - - * * * * | 29 21 13 5 * * * * | - - - - * * * * | 30 22 14 6 * * * * | - - - - * * * * | 31 23 15 7 * * * * - Tiling alignment restrictions subsume FR restrictions because the FR surface base address is the same as the start address. **Register Programming**: FR_MODE - inplace or copy FR_TYPE - type of transformation DSTCD - bpp8, bpp16, bpp32

Bit	Reset	Description
		SRCBA - source base address SRCWIDTH - (width in pixels-1) SRCHEIGHT - (height in lines-1) SRCS - source stride DSTBA - destination base address DSTS - destination stride FR_READWAIT - always set to enable FR inefficency in the following setup: 1. FR_MODE==SQUARE 2. if(FR_TYPE==YFLIP && SRCHEIGHT==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 Or if(FR_TYPE==XFLIP && SRCWIDTH==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 3. Then, the middle 8 lines/pixels, e.g., YFLIP/16bpp/n==3, line8 to line15 will be processed twice. 00 = Disable fast rotate - this turns off the second level clock to fr_rotate engine. Please remember to do so! 01 = src/dst copy mode - two separate buffers 10 = square in place - one buffer 11 = blank 0 = DISABLE 1 = SRC_DST_COPY 2 = SQUARE 3 = BLANK
22:21	0x0	CLIPC: Clipping rectangle control, if clip enable, bit 57, XYTDW, should be cleared. 0x=clipping disabled 10=draw only inside clipping rectangle 11=draw only outside clipping rectangle
9	0x0	ALPSRCORDST: 32-bit blending mode, output alpha selection 0:source alpha, 1:destination alpha 0 = DISABLE 1 = ENABLE
8:4	0x0	ALPTYPE: Alpha blending method FIX: ALPHA blending: Fixed alpha, ALPHA is the value, B5G6R5 * B5G6R5 VCAA: B5G6R5->B5G6R5 PL1BPP: ALPHA blending: Alpha 1 bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero PL2BPP: ALPHA blending: Alpha 2 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero PL4BPP: ALPHA blending: Alpha 4 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero PL8BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1 bit from source B5G5R5A1, alpha (MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4 bits from source A4B4G4R4, alpha (LSB). Dest: B5G6R5. PLS4BPP: ALPHA blending: Alpha 4 bits from source B4G4R4A4, alpha (MSB). Dest: B5G6R5. VCAA: R8G8B8A8->B5G6R5 without reading VCAA plane (surface blit with bpp down convert - implemented in hardware by the VCAA engine; not really a VCAA resolve) PLS8BPP: ALPHA blending: Alpha 8 bits from source/destination (decided by ALPSRCORDST), R8G8B8A8, alpha(MSB). DST: R8G8B8A8 VCAA: R8G8B8A8->R8G8B8A8, alpha has same blending method as RGB PLS8BX: ALPHA blending: Alpha 8 bits from source B8G8R8A8, alpha (MSB). Dest: B5G6R5. (**Restrictions) VCAA: R8G8B8A8->B5G6R5 PLS1BPPAL: ALPHA blending: Alpha 1 bit from source A1B5G5R5, alpha (LSB). Dest: B5G6R5 VCAA: A1B5G5R5->A1B5G5R5

Bit	Reset	Description
		**Restriction PLS8BX alpha blending has the following restrictions: 1. Source/destination addresses have to be in a 128-bit boundary. 2. Destination width has to be a multiple of 4 pixels. 3. Source/Destination strides have to be a multiple of 128 bits. 0 = FIX 1 = PL1BPP 2 = PL2BPP 3 = PL4BPP 4 = PL8BPP 5 = PL44BPP 6 = PLS1BPP 7 = PLS4BPPAL 8 = PLS4BPP 9 = PLS8BPP 10 = PLS8BX 11 = PLS1BPPAL
3	0x0	BEWSWAP: Host port word swap 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 0 = DISABLE 1 = ENABLE

14.5.26 G2SB_CTX3_G2CONTROLMAIN_0

G2 Blit Formats:

(1) G2 Blit size is based on height in lines, width in pixels, stride in bytes, and pixel size (color depth). Unless alpha blending is enabled, G2 does not care about pixel components.

Source color depths supported by G2 Blit:

- color depth same as destination: G2CONTROLMAIN.SRCCD = 1
- monochrome: G2CONTROLMAIN.SRCCD = 0

(source monochrome color determined by G2SRCBGC, G2SRCFGC)

Pattern color depths supported by G2 Blit:

- color depth same as destination: G2PATOS.PATCD = 1
- monochrome: G2PATOS.PATCD = 0

(pattern monochrome color determined by G2PATBGC, G2PATFGC)

Destination color depths supported by G2 Blit:

- 1 byte per pixel: G2CONTROLMAIN.DSTCD = BPP8
- 2 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP16
- 4 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP32

(2) Alpha blending operates on 3 or 4 component pixels of a specified color depth. One of the components may be alpha (A), depending on the alpha blend format, which is only allowed in certain component positions defined below. The only format rule of the remaining components is that they must be in the same source and destination position because the alpha blend engine

does not support component swapping. For example, valid 32BPP formats are: RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888, GRBA_8888->GRBA_8888, etc.

Source formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16:
  - ALPTYPE_PLS1BPP: xxxA_5551 (for example, BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
  - ALPTYPE_PLS4BPP: xxxA_4444 (for example, BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
  - (other) : xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32:
  - (all) : xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

Destination formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16: xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])
- Destination formats supported by G2 alpha blend:

src cd	src format	dst cd	dst format	examples
BPP8	no support	BPP8	no support	
BPP16	xyzA_5551, xyzA_4444, xyz_565	BPP16	xyz_565	BGRA_5551->BGR_565, RGB_565->RGB565
BPP32	xyzA_8888	BPP32	xyzA_8888	RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888
BPP32	xyzA_8888	BPP16	xyz_565	RGBA_8888->RGB_565, BGRA_8888->BGR_565

VCAA Resolve Engine

(1) Format support

srcColor format	srcCoverage format	legal dest format(s)
A1B5G5R5	VCAA	A1B5G5R5
R8G8B8A8	VCAA	R8G8B8A8, B5G6R5
B5G6R5	VCAA	B5G6R5

To program resolve type:

- ALPTYPE == FIX: B5G6R5 -> B5G6R5
- ALPTYPE == PLS1BPPAL: A1B5G5R5 -> A1B5G5R5
- ALPTYPE == PLS8BPP: R8G8B8A8 -> R8G8B8A8
- ALPTYPE == PLS8BX: R8G8B8A8 -> B5G6R5

(2) Color surface programming

- color surface base address: SRCBA
- subsurface start x: SRCX (pixel index)
- subsurface start y: SRCY (line index)
- color surface stride: SRCS (bytes)

The engine supports subsurface resolve. One can imagine a full color buffer of 1024x768, but

```
[pseudo_code]
```

```
      color_surface_start_address_fetch = pcurctx->regs.rG2SRCBA.uSRCBA() +
       (pcurctx->regs.rG2SRCPS.uSRCX() * VCAAState.color_surface_depth) +
       (pcurctx->regs.rG2SRCPS.uSRCY() * VCAAState.color_surface_stride);
```

(3) VCAA surface programming

- VCAA surface base address: PATBA
- VCAA surface stride: PATST (bytes)

```
[pseudo_code]
      vcaa_surface_start_address_fetch = pcurctx->regs.rG2PATBA.uPATBA();
```

Software *MUST* directly program this register to the proper subsurface location in the VCAA surface which corresponds to the SRCX and SRCY programming. The calculation is PATBA = vcaa_base_address + SRCX * (1) + SRCY * PATST;

(4) Resolve surface programming

- resolve window width: DSTWIDTH (pixels)
- resolve window height: DSTHEIGHT (lines)
- resolve output stride: DSTS (bytes)

These values program the width and height of the resolved surface. In the base case,

- DSTWIDTH = color surface width
- DSTHEIGHT = color surface height
- DSTS = resolve surface format bpp * DSTWIDTH

The VCAA engine technically supports DSTX and DSTY (like SRCX and SRCY).

```
  [pseudo_code]
     resolve_surface_start_address_put = pcurctx->regs.rG2DSTBA.uDSTBA() +
       (pcurctx->regs.rG2DSTPS.uDSTX() * resolve_surface_color_depth) +
       (pcurctx->regs.rG2DSTPS.uDSTY() * resolve_surface_color_stride);
```

(5) Resolving a pixel

The coverage surface is 8 bits per pixel surface (C4X4). The surface is initialized to all 1s by 3d.

For a given pixel, we need to look up 4 bits of coverage data. If the data is 0xf, the pixel does not need to be resolved. Otherwise, the pixel needs to be reblended with certain weightings of its neighbors (i.e., we are on the edge of some geometry).

If any coverage bit is 0, the color needs to be resolved. To do so, the following equation is calculated:

```
   color_new = (
                 20*color_old +
                 27*(
                     ((cover_down  == 0) ? color_down  : color_old) +
                     ((cover_right == 0) ? color_right : color_old) +
                     ((cover_left  == 0) ? color_left  : color_old) +
                     ((cover_up    == 0) ? color_up    : color_old)
                    )
               ) / 128;
```

If coverage bit == 0, then use the neighbor color value in the resolve.

If coverage bit == 1, then use the center color value in the resolve.

The resolve is done on a per channel (r/g/b/a) basis:

format convert input channel to 8-bit

perform resolve equation above

format convert to output format

For the resolution of the alpha channel:

- In B5G6R5: no alpha information. do nothing.
- In A1B5G5R5: no resolve. keep the source color alpha bit.
- In R8G8B8A8: normal 8-bit channel resolve

(6) Resolving a pixel on the edge

When on an edge, the VCAA engine treats the coverage bit as a 1. Use the center color value. This holds true even on the edge of the subsurface resolve window when located in the middle of the larger surface

(7) Maximum dimensions

The maximum resolve sub-surface is 4096 pixels wide.

(8) Surface restrictions

The VCAA engine has the following restrictions with respect to input surfaces

- input base address is 128-bit memory word aligned

- input stride is a multiple of 128 bits (16 bytes)

The VCAA engine has the following restrictions with respect to output surfaces:

- output base address is 128-bit memory word aligned

- output stride is a multiple of 128 bits (16 bytes)

(9) Coverage surface layout

The surface is C4X4.

Surface layout (C4X4)

```
(lsb2msb)  C0_C1_C2_C3_X_X_X_X     (X is a don't care.  Qrast inits to 1)
(msb2lsb)  X_X_X_X_C3_C2_C1_C0


    +          o          o
         C0
            C1   Cov(0,1)
       C2
         C3
    o          o          o

      Cov(1,0)   Cov(1,1)

    o          o          o

    +/o = pixel centers
```

In the example above,

```
Cov(0,0) = X_X_X_X_C3_C2_C1_C0
C3       = V_C3(Cov(0,0));
..         ..
C0       = V_C0(Cov(0,0));
#define V_C3(bits) (bits & 0x8)
#define V_C2(bits) (bits & 0x4)
#define V_C1(bits) (bits & 0x2)
#define V_C0(bits) (bits & 0x1)
```

Resolving a pixel

In an ideal world, the coverage would be simply laid out such that when resolving Pixel(x,y), we would need only to fetch Cov(x,y) to get all 4 bits of information.

In actuality, the coverage surface was rearranged to help qrast in performance. To resolve Pixel(x,y) gr2d has to fetch 4 C4X4 bytes [Cov(x,y), Cov(x-1,y), Cov(x,y-1), Cov(x-1,y-1)]. From each C4X4 byte, it extracts one bit which is used in the resolve equations shown in (5).

Need the logical VCAA bits (DRLU) for blend which is spread across four C4X4 coverage bytes

```
      U
  L
    o
       R
   D
```

Where

o = Pixel(x,y)
D = V_C1(Cov(x-1,y))
R = V_C0(Cov(x,y))
L = V_C3(Cov(x-1,y-1))
U = V_C2(Cov(x,y-1))

Offset: 0x101f | Byte Offset: 0x407c | Read/Write: R/W | Reset: 0x00000000 (0bx000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: Pattern Data Select: SRCSEL and PATSEL cannot be both enabled. 0 = SCREEN 1 = MEMORY
28	0x0	DSTDIR: destination direct addressing
27	0x0	SRCDIR: source direct addressing
26:25	0x0	GCSW: Display Switching Window Control (GCSW[1:0]) This parameter controls multi-buffering for Display. x0: At end of current command, do not send signal to Display to switch buffer. 01: two buffers, dstba and dstba_b are used 11: three buffers, dstba, dstba_b, dstba_c are used
24	X	SRCBAS: Source base address select: 0=srcba

Bit	Reset	Description
		1=dstba This is not used for Line Draw and if source data comes from host memory.
23:22	0x0	SRCT: Source transparency enable: 0x=source transparency disabled, 10=mono source background transparency or color source transparency, 11=mono source foreground transparency or inverse color source transparency. NOTE: When source transparency is enabled and SRCCD==0(mono) SRCBGC!=SRCFGC should be satisfied.
21	X	HLMONO: Start from msb or lsb in byte when mono expansion. If HLMONO is 1, bit 0 (the lsb) is the first bit. If HLMONO is 0, bit 7 (the msb) is the first bit
20	X	SRCCD: 0: Source mono 1: Source has same color depth as destination SRCCD==0 xdir/ydir has to be 0
19:18	0x0	DSTT: Destination read transparency enable: 0x=destination read transparency disabled 10=color destination read transparency 11=inverse color destination read transparency. 20 rw INIC Initiate Command (1=initiate command immediately, 0=wait for launch write)
17:16	X	DSTCD: Destination color depth: 00=8-bpp, 01=16-bpp, 10=32-bpp. 11=reserved 0 = BPP8 1 = BPP16 2 = BPP32 3 = RESERVED3
15	0x0	SRCSEL: Source Data Select: 0 = SCREEN 1 = MEMORY
14	0x0	YFLIP: flip y direction to make image upside down or the other way. If YFLIP==1, ROP cannot include destination. 0 = DSIABLE 1 = ENABLE
13	0x0	PATPACK: Pattern Data is in Pack Mode. PATLNGAP in G2PATPACK is the line gap for pattern packed mode If (PATPACK && ~PATSEl), pattern data is packed and from screen, PATMONOW/PATMONOH should be programmed properly to fetch pattern data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
12	0x0	SRCPACK: Source Data is in Pack Mode. SRCLNGAP in G2SRCPACK is the line gap for source packed mode. If (SRCPACK && ~SRCSEL), source data is packed and from screen, SRCMONOW/SRCMONOH should be programmed properly to fetch data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
11	0x0	XYTDW: xy transpose, Line stride DSTS has to be 16-byte aligned if enabled. If XYTDW==1, ROP cannot include destination.
10	0x0	YDIR: 0=incrementing, 1=decrementing. ydir should not be set when source surface has

Bit	Reset	Description
		different color depth as destination surface. SRCCD==0 (mono src) OR PATCD==0 (mono pat), ydir has to be 0
9	0x0	XDIR: 0=incrementing, 1=decrementing. xdir should not be set when source surface has different color depth as destination surface. e.g. ALPTYPE=PLS8BX (32bpp blending with 16bpp), xdir has to be 0 SRCCD==0 (mono src) OR PATCD==0 (mono pat), xdir has to be 0

8	0x0	PATFL: When mono pattern is set, we use mono tile pattern fill. Current pattern client can support a 16x16 bit tile which can be used to generate a larger surface (i.e. tile this surface in the x and y direction) The tile is stored in memory.
		Currently, PATXO and PATYO indicate where to start in the 16x16 tile when expanding the monochrome data. See patxo and patyo comments below for programming. When xdir==1 Patxo = pattern width - (patxo+destination width)&0xF -- For xoffsets y offsets PATYO remains the same when xdir/ydir change. How the tile replication pattern looks based on xdir/ydir (without programming patxo above): xdir=0 ydir=0 xdir=1 ydir=0 +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ // xdir=0 ydir=1 xdir=1 ydir=1 Mono tile is +---+ |^|^| |~~~| +---+
		0 = DISABLE
		1 = ENABLE

7	0x0	PATSLD: BitBlt Solid Pattern Fill: 1=enable. BGC will be used as the color value. 0 = DISABLE 1 = ENABLE
6	0x0	SRCSLD: BitBlt Solid Source Color Fill: 1=enable. FGC will be used as the color value. 0 = DISABLE 1 = ENABLE
5	0x0	ALPEN: BitBlt Alpha Blending, 1=enable. 0=disable, when both Faden and alpen are 1, output=Source*alpha_v + fadoff, alpha_v is decided by alptype
4	0x0	FADEN: BitBlt Source Copy Fade enable, 1=enable (share with mltln), only support 16bpp mode 0 = DISABLE 1 = ENABLE
3	0x0	TEST0BIT: Command finish timing bit 0: 2D command finishes when last data has been pushed to memory write client. 1: 2D command waits memory write client to be idle to finish. 0 = DISABLE 1 = ENABLE
2	0x0	TURBOFILL: fast fill rectangle in 128 bits/clock. Some limitations with this mode:srcsld==1 rop==0xcc, no clipping, no transparency, xdir==0, ydir==0, flip==0, xytdw==0. Results are undefined if the above limitations are not satisfied.
1:0	0x0	CMDT: Command Type: 00=BitBlt 01=Line Draw 10=VCAA 11=reserved When the raise command is executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

14.5.27 G2SB_CTX3_G2ROPFADE_0

Offset: 0x1020 │ Byte Offset: 0x4080 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description

Bit	Reset	Description
31:24	X	FADOFF:
23:16	X	FADCOE:
7:0	0x0	ROP: If YFLIP==1 or XYTDW==1, ROP cannot include destination. Since destination may have been corrupted before reading out.

14.5.28 G2SB_CTX3_G2ALPHABLEND_0

Offset: 0x1021 | Byte Offset: 0x4084 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
31:24	X	ALPHA1V
23:16	X	ALPHA0V
8	0x0	ALPHAINV
7:0	X	ALPHA

14.5.29 G2SB_CTX3_G2CLIPLEFTTOP_0

ClipRect Methods. Top-left bounds are inclusive

Offset: 0x1022 | Byte Offset: 0x4088 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPT
14:0	X	CLIPL

14.5.30 G2SB_CTX3_G2CLIPRIGHTBOT_0

Bottom-right bounds are exclusive

Offset: 0x1023 | Byte Offset: 0x408c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPB
14:0	X	CLIPR

14.5.31 G2SB_CTX3_G2PATPACK_0

Pattern Packed Mode

Pattern methods: G2PATPACK should be used to specify the line gap. Use G2PATPACK_SIZE to program height and width PACK is generally only useful with narrow monochrome surfaces

> **Note:** PACK is not officially supported when the surface is tiled.

Offset: 0x1024 | Byte Offset: 0x4090 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

14.5.32 G2SB_CTX3_G2PATPACK_SIZE_0

Pattern Packed Mode

Packed mode, pattern data line gap. Byte

G2PATPACK_SIZE -- extension of G2PATPACK, this register only holds the size of the packed data

> **Note:** PACK is not officially supported when the surface is tiled.

Offset: 0x1025 | Byte Offset: 0x4094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	PATMONOH: Pattern mono data height in packed mode, byte
15:0	X	PATMONOW: Mono data width in packed mode, byte

14.5.33 G2SB_CTX3_G2PATBA_0

Offset: 0x1026 | Byte Offset: 0x4098 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBA: If (PATFL==1){ It has to be 16-byte aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

14.5.34 G2SB_CTX3_G2PATOS_0

Offset: 0x1027 | Byte Offset: 0x409c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00xxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	PATYO: y offset for mono tile pattern fill. see PATFL
27:24	X	PATXO: x offset for mono tile pattern fill. see PATFL
22:21	0x0	PATT: Mono pattern transparency enable: 0x=pattern transparency disabled 10=mono pattern background transparency or color pattern transparency 11=mono pattern foreground transparency or inverse color pattern transparency NOTE: When pattern transparency is enabled and PATCD==0(mono) PATBGC!=PATFGC should be satisfied.
16	X	PATCD:0 mono1 same as dstcd PATCD==0 xdir/ydir has to be 0
15:0	X	PATST: stride

14.5.35 G2SB_CTX3_G2PATBGC_0

Offset: 0x1028 | Byte Offset: 0x40a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

14.5.36 G2SB_CTX3_G2PATFGC_0

Offset: 0x1029 | Byte Offset: 0x40a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

14.5.37 G2SB_CTX3_G2PATKEY_0

Offset: 0x102a | Byte Offset: 0x40a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

14.5.38 G2SB_CTX3_G2DSTBA_0

Offset: 0x102b | Byte Offset: 0x40ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA: Destination base address (byte address)

14.5.39 G2SB_CTX3_G2DSTBA_B_0

Offset: 0x102c | Byte Offset: 0x40b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_B: Destination base address (byte address) only usable in hardware trigger mode by enable gcsw

14.5.40 G2SB_CTX3_G2DSTBA_C_0

Offset: 0x102d | Byte Offset: 0x40b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_C: Destination base address (byte address)

14.5.41 G2SB_CTX3_G2DSTST_0

Offset: 0x102e | Byte Offset: 0x40b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

14.5.42 G2SB_CTX3_G2SRCPACK_0

Source Data Packed Mode

Surface methods: G2SRCPACK should only be used to specify the line gap. Use G2SRCPACK_SIZE to program height and width

Offset: 0x102f │ Byte Offset: 0x40bc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

14.5.43 G2SB_CTX3_G2SRCPACK_SIZE_0

Source Data Packed Mode

In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, the required width/stride is 16 byte aligned. Packed mode - source mono data line gap

G2SRCPACKS_SIZE -- extension of G2SRCPACK, this register only holds the size of the packed data

Offset: 0x1030 │ Byte Offset: 0x40c0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCMONOH: Packed mode - source mono data height in bytes
15:0	X	SRCMONOW: Packed mode - source mono data width in bytes

14.5.44 G2SB_CTX3_G2SRCBA_0

SRC Base Address

Offset: 0x1031 │ Byte Offset: 0x40c4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA: Source base address (byte address)

14.5.45 G2SB_CTX3_G2SRCBA_B_0

SB only

Offset: 0x1032 │ Byte Offset: 0x40c8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA_B: This parameter specifies the start address of source image stored in the image buffer memory. In 4:2:0 format mode, this image block accommodates for Y-image. This address specifies byte position; however, bits [2:0] are restricted with respect to the data formats to fit multiple pixels in one memory word (8 bytes). For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format. Since one Y pixel takes 8 bits, all 8 byte positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there are no restrictions on this value.)

14.5.46 G2SB_CTX3_G2SRCST_0

Offset: 0x1033 | Byte Offset: 0x40cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS: Source Stride coordinate (bytes) with respect to SRCBA. In order to fit multiple pixels in one memory word (8 bytes), bits [2:0] are restricted with respect to the data formats. For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format.

14.5.47 G2SB_CTX3_G2SRCBGC_0

srcColors

Offset: 0x1034 | Byte Offset: 0x40d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

14.5.48 G2SB_CTX3_G2SRCFGC_0

Offset: 0x1035 | Byte Offset: 0x40d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

14.5.49 G2SB_CTX3_G2SRCKEY_0

Offset: 0x1036 | Byte Offset: 0x40d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

14.5.50 G2SB_CTX3_G2SRCSIZE_0

Offset: 0x1037 | Byte Offset: 0x40dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	SRCHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	SRCWIDTH

14.5.51 G2SB_CTX3_G2DSTSIZE_0

Offset: 0x1038 | Byte Offset: 0x40e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	DSTHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	DSTWIDTH: In 2D mode, the largest number can be programmed is 0x7fef

14.5.52 G2SB_CTX3_G2SRCPS_0

ImageBlit Methods

Offset: 0x1039 | Byte Offset: 0x40e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCY
15:0	X	SRCX:SRCX[2:0] are ignored in SRCCD==0 (mono expansion). The first bit of the first byte (bit 7 if HLMONO==0, or bit 0 if HLMONO==1) always expands to DSTX,DSTY

14.5.53 G2SB_CTX3_G2DSTPS_0

Offset: 0x103a | Byte Offset: 0x40e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	DSTY
15:0	X	DSTX: NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero

14.5.54 G2SB_CTX3_G2CBDES_0

Circular Buffer

Offset: 0x103b | Byte Offset: 0x40ec | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TOPCLIP: top clipping at the first buffer, (buffer start) refer to TOP_CBLINE in G2CBDES2 0 = DISABLE 1 = ENABLE
30:16	X	CBLINE: vertical line number in one buffer
7:0	X	CBCOUNT: This specifies the number of buffers in circular buffer feature

14.5.55 G2SB_CTX3_G2CBSTRIDE_0

Offset: 0x103c | Byte Offset: 0x40f0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	X	CBUVSTRIDE: Chroma Buffer Stride default is half of luma 00= 1/2 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 2 bytes. 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 4 bytes. 1x= Reserved 0 = CBS2X 1 = CBS1X 2 = CBS4X
23:0	X	CBSTRIDE: Video Buffer Luma (or RGB) Buffer Stride. This is luma buffer stride (in bytes)

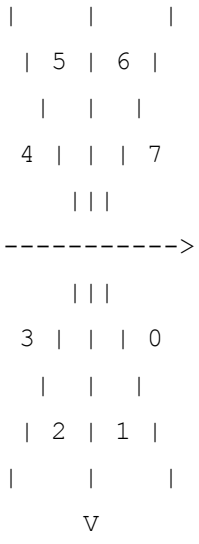
14.5.56 G2SB_CTX3_G2LINESETTING_0

Line Methods

Offset: 0x103d | Byte Offset: 0x40f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	OCTANTS: 000: octant 0 001: octant 1 010: octant 2 011: octant 3 100: octant 4 101: octant 5 110: octant 6 111: octant 7
28	X	LINEUSEOCTANT: use OCTANTS in G2LINEDELTAN register instead of MAJOR LINEXDIR LINEYDIR
27	X	DROPLASTP: draw last pixel or not
26	X	LINEYDIR
25	X	LINEXDIR
24	X	MAJOR: 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

14.5.57 G2SB_CTX3_G2LINEDELTAN_0



Offset: 0x103e | Byte Offset: 0x40f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

14.5.58 G2SB_CTX3_G2LINEDELTAM_0

Offset: 0x103f | Byte Offset: 0x40fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

14.5.59 G2SB_CTX3_G2LINEPOS_0

Offset: 0x1040 | Byte Offset: 0x4100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LINEYPOS
15:0	X	LINEXPOS

14.5.60 G2SB_CTX3_G2LINELEN_0

Offset: 0x1041 | Byte Offset: 0x4104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

14.5.61 G2SB_CTX3_G2CSCFOURTH_0

Offset: 0x1042 | Byte Offset: 0x4108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:16	X	G2V: Multiplier for G for V generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.368 (decimal) or 0x12F. For any other combination, this parameter is ignored.
8:0	X	G2U: Multiplier for G for U generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.291 (decimal) or 0x125. For any other combination, this parameter is ignored.

14.5.62 G2SB_CTX3_G2SRCST_B_0

Offset: 0x1043 | Byte Offset: 0x410c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

14.5.63 G2SB_CTX3_G2UVSTRIDE_0

Offset: 0x1044 | Byte Offset: 0x4110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

14.5.64 G2SB_CTX3_G2CBDES2_0

Circular Buffer Controller 2

Offset: 0x1045 | Byte Offset: 0x4114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

14.5.65 G2SB_CTX3_G2TILEMODE_0

Offset: 0x1046 | Byte Offset: 0x4118 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx0xxx0xxx0xxx0xxx0xxx0)

Bit	Reset	Description
20	0x0	DST_WR_TILE_MODE: destination surface 0 = LINEAR 1 = TILED
16	0x0	DST_RD_TILE_MODE: Same as destination write unless DST_RD_WR_SEP (not supported) 0 = LINEAR 1 = TILED
12	0x0	PAT_UV_TILE_MODE:UNUSED 0 = LINEAR 1 = TILED
8	0x0	PAT_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED
4	0x0	SRC_UV_TILE_MODE:UV surface, ignored in RGB mode 0 = LINEAR 1 = TILED
0	0x0	SRC_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED

14.5.66 G2SB_CTX3_G2PATBASE_0

Offset: 0x1047 | Byte Offset: 0x411c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PAT_BASE: pattern base address in tile mode, PATBA is the linear address where pixel start

14.5.67 G2SB_CTX3_G2SRCBA_SB_SURFBASE_0

SB_SURFBASE Registers

These registers need only be programmed when using SB. They point to the base of the various source and destination surfaces. Technically, they are only needed when tiling is enabled, but there is no harm in always programming them.

Their counterpart registers (_SB_SURFBASE with SB_SURFBASE stripped off) indicate the location of the first pixel to be sourced or written within the surface.

This register exists to mimic the X, Y (SRCX, SRCY) functionality found in the BitBlt engine.

For example, to get at pixel X, Y:

SRCBA = SRCBA_SB_SURFBASE + Y*stride + X*Bpp

Offset: 0x1048 | Byte Offset: 0x4120 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRC_ADDR: surface address corresponding to G2SRCBA: -base of interleaved sources (RGB, YUV) -base of Y plane. Only used by the StretchBlit Engine

14.5.68 G2SB_CTX3_G2DSTBA_SB_SURFBASE_0

Offset: 0x1049 | Byte Offset: 0x4124 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_ADDR: surface address corresponding to G2DSTBA. Only used by the StretchBlit Engine

14.5.69 G2SB_CTX3_G2DSTBA_B_SB_SURFBASE_0

Offset: 0x104a | Byte Offset: 0x4128 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_B_ADDR: surface address corresponding to G2DSTBA_B. Only used by the StretchBlit Engine, and G2CONTROLSB.DBSEL() is enabled

14.5.70 G2SB_CTX3_G2VBA_A_SB_SURFBASE_0

Offset: 0x104b | Byte Offset: 0x412c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	V_ADDR: surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of V plane. Only used by the StretchBlit Engine

14.5.71 G2SB_CTX3_G2UBA_A_SB_SURFBASE_0

Offset: 0x104c | Byte Offset: 0x4130 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	U_ADDR: surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of U plane. Only used by the StretchBlit Engine

14.6 G2SB CTX 4 Registers

14.6.1 G2SB_CTX4_INCR_SYNCPT_0

Memory Controller Tiling Definitions

Offset: 0x5000 | Byte Offset: 0x14000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE

Bit	Reset	Description
		3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	INDX: syncpt index value

14.6.2 G2SB_CTX4_INCR_SYNCPT_CNTRL_0

Offset: 0x5001 | Byte Offset: 0x14004 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

14.6.3 G2SB_CTX4_INCR_SYNCPT_ERROR_0

Offset: 0x5002 | Byte Offset: 0x14008 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

14.6.4 G2SB_CTX4_G2CLASSCHANNEL_REGONLY_0

This register can be accessed by both register read/write and command FIFO in order not to break exsting code , keep suffix REGONLY.

2D syncpt behavior

- Each 2D context has its own syncpt client.
- OP_DONE increments when 2D's memory write client is idle, meaning all preceding operations are complete.
- RD_DONE behaves like OP_DONE
- REG_WR_SAFE behaves like IMMEDIATE meaning a syncpt increment returns immediately and does not indicate that a register write will NOT block.
- The implications are that 2D's host FIFO can fill and block access to 2D, which will indeterminately block a direct CPU access.

Offset: 0x5008 | Byte Offset: 0x14020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID

Bit	R/W	Reset	Description
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

14.6.5 G2SB_CTX4_G2TRIGGER_0

Control methods, since they are not programmed often, put them at the beginning

Offset: 0x5009 | Byte Offset: 0x14024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

14.6.6 G2SB_CTX4_G2TRIGGER1_0

Offset: 0x500a | Byte Offset: 0x14028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

14.6.7 G2SB_CTX4_G2TRIGGER2_0

Offset: 0x500b | Byte Offset: 0x1402c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

14.6.8 G2SB_CTX4_G2CMDSEL_0

Offset: 0x500c | Byte Offset: 0x14030 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxx0xxxxxxxx00x0x0000000xxxx)

Bit	R/W	Reset	Description
28	RW	0x0	PRIORITY: indicates priority of the context, note VI-triggered contexts take priority over HIGH 0 = LOW 1 = HIGH
27:25	RW	X	LINKVAL
24	RW	0x0	LINKEN: Test purpose
23:16	RW	X	BUFFER_INDEX
15	RW	0x0	FRAME_END
14	RW	0x0	FRAME_START: host writes this bit to tell which buffer is ready
10	RW	0x0	CLIP_SOURCE_TOP_BOTTOM: Reserved for hardware test, Software should not use this bit. 0 = DISABLE 1 = ENABLE
9:8	RW	0x0	G2OUTPUT: 00: to memory 01: to EPP 0 = MEMORY 1 = EPP 2 = RESERVED2 3 = RESERVED3

Bit	R/W	Reset	Description
7	RW	0x0	CBSBDISABLE
6	RW	0x0	HOSTTRIGGER: host trigger the command, host also need program circular buffer related data
5	RW	0x0	VITRIGGER:VI trigger enable, can be circular buffer or full frame
4	RW	0x0	CBENABLE: circular buffer feature enable 0 = DISABLE 1 = ENABLE
0	RO	X	SBOR2D: 0 = G2 1 = SB

14.6.9 G2SB_CTX4_G2RAISE_0

Offset: 0x500d │ Byte Offset: 0x14034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE: Push back to read FIFO when all commands in the channel are done.

14.6.10 G2SB_CTX4_G2HOSTSET_0

Offset: 0x500f │ Byte Offset: 0x1403c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	HSTFPXL: specifies the first pixel location (in byte) in the first source data entry. In mono mode pixel should always start in byte boundary
3:0	0x0	HSTLNGAP: specifies distance (in byte) from last pixel of a line to the first pixel of next line. In mono mode, the value is the nearest byte number. e.g., if gap is 3 bits, HSTLNGAP should be 1 (byte), if gap is 9 bits, HSTLNGAP should be 2 (byte)

14.6.11 G2SB_CTX4_G2HOSTFIFO_0

Offset: 0x5010 │ Byte Offset: 0x14040 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

14.6.12 G2SB_CTX4_G2VDDA_0

Offset: 0x5011 │ Byte Offset: 0x14044 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VDSTEP: Vertical Scaling DDA. Reset value: xxxx-xxxxh. Vertical DDA Step (Increment) Value (VDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for vertical scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_height-1-VDTINI) / (Actual_destination_height-1). Truncate the remaining bits to keep the 12-bit fraction. Since we have to meet (Actual_source_height-1)*1.0 >= (Actual_destination_height - 1)*VDSTEP + VDTINI. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half line-spacing for up to

Bit	Reset	Description
		approximately 2000 target lines. For example, image expansion from 240 lines to 400 lines VDSTEP[17:0] = 19'b00_0000_1001_1001_1010 and image contraction from 240 lines to 150 lines VDSTEP[17:0] = 19'b00_0001_1001_1001_1010.

14.6.13 G2SB_CTX4_G2VDDAINI_0

Offset: 0x5012 | Byte Offset: 0x14048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	VDBINI: Vertical DDA Initial Value for Bottom-Field (VDBINI[7:0]) (see description above)
7:0	X	VDTINI: Vertical Scaling DDA Initial Values. Vertical DDA Initial Value for Top-Field (VDTINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for vertical scaling. Given a pair of consecutive source image lines representing positions 0.0 and 1.0, a destination line-image will be created at any position in [0.0, 1.0), that is specified by the fraction part of the vertical DDA. The 8-bit initial fraction value specifies the position of the first output (destination) line-image located between the first and the second input (source) line images, respectively at position 0.0 and 1.0. This parameter provides a way to compensate relative displacement of top and bottom fields of a source image. Suppose, 2-to-1 interlaced original video field images #1 and #2 are in the image buffer, to be displayed in an overlay window. Assume that the #1 field image is the top-field. Using VDTINI = 8'hC0 for the top-field image and VDBINI = 8'h40 for the bottom-field image, for example, the two images from #1 and #2 fields get mapped to identically positioned destination images in the overlay window.

14.6.14 G2SB_CTX4_G2HDDA_0

Offset: 0x5013 | Byte Offset: 0x1404c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HDSTEP: Horizontal Scaling DDA. Reset value: xxxx-xxxxh. Horizontal DDA Step (Increment) Value (HDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for horizontal scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_width-1-HDINI) / (Actual_destination_width-1) Truncate the remaining bits to keep the 12-bit fraction. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half pixel-spacing for up to approximately 2000 destination pixels. For example, image expansion from 720 pixels to 800 pixels HDSTEP[17:0] = 19'b000_0000_1110_0110_0111 and image contraction from 720 pixels to 250 pixels HDSTEP[17:0] = 19'b000_0010_1110_0001_0101.

14.6.15 G2SB_CTX4_G2HDDAINILS_0

Offset: 0x5014 | Byte Offset: 0x14050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	HDINI: Horizontal Scaling DDA Initial Value. Horizontal DDA Initial Value (HDINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for horizontal scaling. Given a group of six consecutive source pixels that the two pixels at the center representing positions 0.0 and 1.0, a destination pixel will be created at any position in [0.0, 1.0), that is specified by the fraction part of the horizontal DDA. The 8-bit initial fraction value specifies the position of the first output (destination) pixel located between the first and the second input (source) pixels, respectively at position 0.0 and 1.0.For horizontal scaling this value may normally be set to 0.

14.6.16 G2SB_CTX4_G2CSCFIRST_0

The next 3 registers contain 8 parameters used by the YCbCr (YUV) to RGB color space conversion logic. This logic can also be used for RGB gain/gamma correction in RGB to RGB modes, and for luma gain in YUV to YUV modes.

Two new parameters (G2U, G2V) have been added for RGB to YUV conversion. These are in address 0x38. The equations have the form:

if (yuv2rgb)

(r,g,b) = matrix*(y+sbreg_yos,u,v)

matrix: sbreg_cyx, sbreg_cur, sbreg_cvr,
sbreg_cyx, sbreg_cug, sbreg_cvg,
sbreg_cyx, sbreg_cub, sbreg_cvb,

else if (rgb2rgb)

(r,g,b) = matrix*(r,g+sbreg_yos,b)

matrix: sbreg_cvr, 0, sbreg_cur,
sbreg_cvg, sbreg_cyx, sbreg_cug,
sbreg_cvb, 0, sbreg_cub,

else if (yuv2yuv)

(y,u,v) = matrix*(y,u,v)

matrix: sbreg_cub, 0, sbreg_cvb,
sbreg_cug, sbreg_cyx, sbreg_cvg,
sbreg_cur, 0, sbreg_cvr,

else if (rgb2yuv)

(y,u,v) = matrix*(r,g,b) + (sbreg_yos,0,0)

matrix: sbreg_cvb, sbreg_cyx, sbreg_cub,
sbreg_cvg, sbreg_g2u, sbreg_cug,
sbreg_cvr, sbreg_g2v, sbreg_cur,

Offset: 0x5015 │ Byte Offset: 0x14054 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	YOS:Y-Offset (YOFFSET[7:0]) for YUV generation This parameter consists of 8-bit 2's complement in the range [-128,127]. For YUV->RGB, the recommended value is -16 (decimal) or 0xF0. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV the recommended value is +16 (decimal) or 0x10
21:12	X	CVR: multiplier for V/R for V or R generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 1.5960 (decimal) or 0x0CCIf source data is in RGB format, this parameter may be used as gain adjustment for R component. For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038.
9:0	X	CUB: multiplier for U/B for Y or B generation. Consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 2.0172 (decimal) or 0x102. If source data is in RGB format, this parameter may be used as gain adjustment for B component. This register changes precision when doing RGB to YUV conversion (SIFMT=1xxx, DIFMT=0xxx). CUB becomes s1.8 and the recommended value is +0.098 or 0x019.

14.6.17 G2SB_CTX4_G2CSCSECOND_0

Offset: 0x5016 | Byte Offset: 0x14058 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	CYX: Multiplier for Y/G (G gain).This positive-only parameter consists of 8-bit magnitude (1.7). For YUV->YUV, the recommended value is 1.1644 (decimal) or 0x95. For YUV->YUV or RGB->RGB, this may be used as gain adjustment for Y or G component. For RGB->YUV, the recommended value is +0.504 (decimal) or 0x041
21:12	X	CUR: Multiplier for U/B for V or R generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB normally this parameter is programmed to 0.0. This parameter takes non-zero value if hue is rotated. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.071 (decimal) or 0x209.
8:0	X	CUG: multiplier for U/B for U or G generation. Consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB, the recommended value is -0.3918 (decimal) or 0x132. For RGB->RGB, this parameter should be set to 0. For YUV->YUV, this parameter should be set to 1 (0x080). For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038.

14.6.18 G2SB_CTX4_G2CSCTHIRD_0

Offset: 0x5017 | Byte Offset: 0x1405c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	CVB: multiplier for V/R for Y or B generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This coefficient takes non-zero value if hue is rotated. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV the recommended value is +0.257 (decimal) or 0x021
8:0	X	CVG: multiplier for V/R for U or G generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB the recommended value is -0.8130 (decimal) or 0x168. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.148 (decimal) or 0x113

14.6.19 G2SB_CTX4_G2CMKEYL_0

When key generation is enabled, the value of this register is the lower color/chroma limit and G2CMKEYU is the upper color/chroma limit for component pixel data. Three component signal values (YCrCb/YUV or RGB) of every output pixel are compared to a set of three ranges that are specified by three pairs of lower and upper color/chroma key values.

If the Key Polarity is 0, the Key is set to 1 only when all three component values of an input pixel are respectively within the set of ranges (inclusive of the limit values), else the Key is set to 0.

If the Key Polarity is 1, the Key is set to 1 when any of the three component values of an input pixel are outside the set of ranges, else the Key is set to 0.

StretchBLT, Color/Chroma Key Lower Limit Register

Offset: 0x5018 | Byte Offset: 0x14060 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRL:R or Cr Color Chroma Key Lower Limit (CKRL[7:0]) Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGL:G or Cb Color Chroma Key Lower Limit (CKGL[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.

Bit	Reset	Description
7:0	X	CKBL: B or Y Color/Chroma Key Lower Limit (CKBL[7:0])

14.6.20 G2SB_CTX4_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0x5019 │ Byte Offset: 0x14064 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRU: R or Cr Color Chroma Key Upper Limit (CKRU[7:0]) Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGU: G or Cb Color/Chroma Key Upper Limit (CKGU[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity the minimum signal level to the maximum signal level.
7:0	X	CKBU: This is B or Y color/chroma key upper limit value.

14.6.21 G2SB_CTX4_G2UBA_A_0

StretchBLT, Source Image Data U base address A-buffer

Offset: 0x501a │ Byte Offset: 0x14068 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SU1SA: Start Address of Source U-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.6.22 G2SB_CTX4_G2VBA_A_0

Offset: 0x501b │ Byte Offset: 0x1406c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SV1SA: Start Address of Source V-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.6.23 G2SB_CTX4_G2SBFORMAT_0

Offset: 0x501c │ Byte Offset: 0x14070 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12:8	X	DIFMT: Destination Image Data Format This parameter defines the data format of destination output. There are two groups of data formats, RGB and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement. 00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 01000: bpp16 5-6-5 {R[4:0], G[5:0], B[4:0]} 01001: RESERVED 01010: RESERVED 01011: RESERVED 01100: bpp16 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]} 01101: RESERVED 01110: R8G8B8A8 01111: B8G8R8A8 1xxxx: RESERVED 0 = U8Y8V8Y8_OB 1 = Y8U8Y8V8_OB 2 = Y8V8Y8U8_OB 3 = V8Y8U8Y8_OB 4 = U8Y8V8Y8_TC 5 = Y8U8Y8V8_TC 6 = Y8V8Y8U8_TC 7 = V8Y8U8Y8_TC 8 = B5G6R5 9 = RESERVED9 10 = RESERVED10 11 = RESERVED11 12 = B5G6R5BS 13 = RESERVED13 14 = R8G8B8A8 15 = B8G8R8A8 16 = RESERVED16 17 = RESERVED17 18 = RESERVED18 19 = RESERVED19 20 = RESERVED20 21 = RESERVED21 22 = RESERVED22 23 = RESERVED23 24 = RESERVED24 25 = RESERVED25 26 = RESERVED26 27 = RESERVED27 28 = RESERVED28 29 = RESERVED29 30 = RESERVED30 31 = RESERVED31
4:0	X	SIFMT: This parameter defines the data format of source input. There are two groups of data formats, RGB format and YCbCr (YUV) format. CbCr (UV) components may be represented in

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>
either offset binary or 2's complement.
00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
01000: B5G6R5 5-6-5 {R[4:0], G[5:0], B[4:0]}
01001: RESERVED
01010: RESERVED
01011: RESERVED
01100: B5G6R5 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]}
01101: RESERVED
01110: R8G8B8A8
01111: B8G8R8A8
1xxxx: RESERVED
StretchBlit Inputs: RGB inputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV420 input = {YUV420 is converted into 4:2:2 UYVY via memory client} YUV422 inputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}
StretchBlit Outputs: RGB outputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV422 outputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}
StretchBlit Input/Output Rules
<table>
<tr><th>src format</th><th>internal sb format</th><th>dst format</th></tr>
<tr><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td><td>R8G8B8A8</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
</table>
StretchBlit color space converter supports YUV->RGB, YUV->YUV (gain) and RGB->RGB (gain). There is no support for RGB->YUV.
0 = U8Y8V8Y8_OB
1 = Y8U8Y8V8_OB
2 = Y8V8Y8U8_OB
3 = V8Y8U8Y8_OB
4 = U8Y8V8Y8_TC
5 = Y8U8Y8V8_TC
6 = Y8V8Y8U8_TC
7 = V8Y8U8Y8_TC
8 = B5G6R5
9 = RESERVED9
10 = RESERVED10
11 = RESERVED11
12 = B5G6R5BS
13 = RESERVED13
14 = R8G8B8A8
15 = B8G8R8A8
16 = RESERVED16
17 = RESERVED17
18 = RESERVED18
19 = RESERVED19
20 = RESERVED20
21 = RESERVED21
</td></tr>
</table>

Bit	Reset	Description
		22 = RESERVED22 23 = RESERVED23 24 = RESERVED24 25 = RESERVED25 26 = RESERVED26 27 = RESERVED27 28 = RESERVED28 29 = RESERVED29 30 = RESERVED30 31 = RESERVED31

14.6.24 G2SB_CTX4_G2CONTROLSB_0

Offset: 0x501d │ Byte Offset: 0x14074 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000xxx)

Bit	Reset	Description
31	0x0	DISDW: Output destination writes go either to image memory or the EPP. 0: Output data is sent to memory 1: YUV or RGB data is passed directly to the EPP module, and no destination writes take place. 0 = GOTO_IMAGE_BUFFER 1 = GOTO_EPP
30	0x0	ENDITH: Enable Dithering (ENDITH). For 16 bit RGB output modes, the LSB of the color components can be modified by adding a variable residual value that will reduce the banding artifacts that can appear on the display. 0: Normal operation 1: Enable Dithering 0 = DISABLE 1 = ENABLE
28	0x0	KPOL: Key Signal Polarity (KPOL). Color/Chroma key signal is generated by comparing source input pixel color to a range of a color specified by lower and upper limit values. The key signal is interpreted in two ways, depending on which one of video and graphics images is foreground (and the other is background). This is effective only if Key signal generator is enabled. (see G2CMKEYL, G2CMKEYU) 0: Key signal is set to 1 when source pixel is within the lower and upper limit color range. 1: Key signal is set to 1 when source pixel is outside the lower and upper limit color range. 0 = WITHIN_BOUNDS 1 = OUTSIDE_BOUNDS
27	0x0	KEYEN: Key Signal Generator Enable (KEYEN) Key signal generator generates either chroma key signal (from YCbCr signal) or color key signal (from RGB signal) 0: Key signal generator is disabled. 1: Key signal generator is enabled. 0 = DISABLE 1 = ENABLE
26	0x0	DBSEL: StretchBLT Destination Buffer Selection (DBSEL) StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the destination video area. This bit selects one of the two buffering blocks to which this StretchBLT command delivers the destination image. The two buffering memory blocks are called A and B. 0: Destination image goes to ‘A’ buffer. 1: Destination image goes to ‘B’ buffer. 0 = DST_A 1 = DST_B

Bit	Reset	Description
25	0x0	SBSEL: StretchBLT Source Buffer Selection (SBSEL) StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the source video area. This bit selects one of the two buffering blocks from which this StretchBLT command receives the source image. The two buffering memory blocks are called source-A and source-B. 0: Source image comes from ‘source-A’ buffer. 1: Source image comes from ‘source-B’ buffer. 0 = SRC_A 1 = SRC_B
24	0x0	SITYPE: StretchBLT Source Type (SITYPE). This bit identifies two types of source images. If source image is 2-to-1 interlaced and StretchBLT processes either one of the two interlaced field-images, physical (positional) displacement between the two interlaced fields must be taken into account. One field-image that is placed higher in position than the other field-image is called top-field and the other is called bottom-field. StretchBLT processing has to lower the top-field (or raise the bottom-field) to match the corresponding two target images in the overlay window (that is progressively scanned) right at the same position. If source image is full frame-image obtained from two interlaced field-images, its type is ‘top-field’. If source images are progressively scanned, the type identification is not significant and they may be designated either one of the two types 0: Source image is ‘top-field’. 1: Source image is ‘bottom-field’. 0 = TOP_FIELD 1 = BOTTOM_FIELD
23	0x0	RANGEREDFRM:In the VC-1 specification, when the value of RANGEREDFRM variable (1-bit) for a picture is equal to 1, the picture shall be scaled up according to the following equation:Y = clip(((Y-128)*2) + 128);Cb = clip(((Cb-128)*2) + 128);Cr = clip(((Cr-128)*2) + 128);The clip operator limits the output to [0, 255].The input is also limited to [0, 255]. In the VC-1 nomenclature, the output (Y, Cb, Cr) tuple corresponds to the 'decoded' picture. The input (Y, Cb, Cr) tuple corresponds to the 'reconstructed' picture. The above equations create the decoded picture while keeping the reconstructed picture intact.Only YUV _OB formats are supported with range reducation enabled, not the YUV _TC or RGB formats The YUV data coming from the video frame buffers (reconstructed picture) in the SDRAM was scaled up using the above equations and given to the display. the value of 1-bit wide RANGEREDFRM variable was configured in the DVO per frame and the frame was scaled whenever RANGEREDFRM == 1. 0 = DISABLE 1 = ENABLE
22:20	0x0	HFTYPE: StretchBLT Horizontal Filter Mode (HFTYPE[2:0]) The six-tap horizontal interpolation filter can be operated in various operation modes. For the image expansion, it should be programmed as a pure 6-tap interpolator. For the image contraction, it can work as partly lowpass filter and partly interpolater with varying degree depending the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of horizontal filtering. 000: Pure interpolation filter. 001, 010, 011, 100, 101, 110: mix of interpolation and low pass filters 111: DISABLE. 0 = INTERP 1 = LPF1 2 = LPF2 3 = LPF3 4 = LPF4 5 = LPF5 6 = LPF6 7 = DISABLE

Bit	Reset	Description
19	0x0	DISCSC: enabled color space converter by default 0 = ENABLE 1 = DISABLE
18	0x0	VFEN: StretchBLT Vertical Filter Enable (VFEN) Vertical filter shall be disabled if source images come from host CPU via CPU Read-FIFO. (SISEL) Vertical filter may be disabled to save some memory bandwidth but this will likely result in degradation of image quality. This option may also be used in the case where source image is progressive scanning and there is no vertical scaling. 0 Vertical filter is disabled.1 Vertical filter is enabled. 0 = DISABLE 1 = ENABLE
17:16	0x0	VFTYPE: StretchBLT Vertical Filter Mode (VFTYPE[1:0]). This bit is effective when the Vertical Filter is enabled. (bit 18) The two-tap vertical interpolation filter can be operated in various modes. For the image expansion, it should be programmed as a pure 2-tap interpolator. For the image contraction, it can work as partly averager and partly interpolator with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of vertical filtering. 00: Pure interpolation filter. 01: 25% averager, 75% interpolator. 10: 50% averager, 50% interpolator. 11: 100% averager. 0 = INTERP 1 = AVG25_INTERP75 2 = AVG50_INTERP50 3 = AVG
9:8	X	UVST: 00= 1/2 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 2 bytes. 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 4 bytes. 11= use uvstride. Note: Use care when using a tiled surface, since 1/2 or 1/4 of luma stride may yield an invalid pitch (stride). 0 = UVS2X 1 = UVS1X 2 = UVS4X 3 = UVS_G2UVSTRIDE
7	0x0	ENAHF: enable horizontal alpha filtering if disabled, use the alpha value of third tap for output pixelU,V line stride in 4:2:0 Format. 0 = DISABLE 1 = ENABLE
6	0x0	ENAVF:enable alpha vertical filter 0 = DISABLE 1 = ENABLE
5	0x0	IMODE: Source (input) Data Mode (IMODE). This parameter defines the data mode of source input. 0: Multiplexed Mode (data format is specified by SIFMT[2:0]). One block of source data in the Image Buffer memory contains multiplexed component signals. 1: Planar mode, 4:2:0 YUV (SIFMT[2] specifies data format of U/V components). Three blocks of source data in the Image Buffer memory contain separate Y, U, and V component signals. For a circular buffer input to gr2d, the input format cannot be planar. It must be multiplex, i.e., no YUV420 planar circular buffer input to gr2d. 0 = MULTIPLEX 1 = PLANAR

Bit	Reset	Description
4	0x0	YUV422PLANAR: Source data is yuv422 planar valid when imode==PLANAR
3	0x0	YUV422ROTATION: YUV422 planar rotated.

14.6.25 G2SB_CTX4_G2CONTROLSECOND_0

Offset: 0x501e │ Byte Offset: 0x14078 │ Read/Write: R/W │ Reset: 0x00000000 (0bxx000000x00xxxx000xxxx000000000x)

Bit	Reset	Description
29	0x0	FR_READWAIT: Fast Rotate wait for reads. Enabling this bit forces FR to wait for the reads to be stored in the data return FIFO before sending writes out to the same block. Always set to ENABLE. This field is debug only. 0 = DISABLE 1 = ENABLE
28:26	0x0	FR_TYPE: Fast Rotate type 000 = FLIP_X 001 = FLIP_Y 010 = TRANS_LR (mirrors about diagonal. Diagonal runs from upper left to lower right) 011 = TRANS_RL (mirrors about diagonal. Diagonal runs from upper right to lower left) 100 = ROT_90 (counterclockwise by 90 degrees) 101 = ROT_180 110 = ROT_270 (clockwise by 90 degrees) 111 = IDENTITY 0 = FLIP_X 1 = FLIP_Y 2 = TRANS_LR 3 = TRANS_RL 4 = ROT_90 5 = ROT_180 6 = ROT_270 7 = IDENTITY
25:24	0x0	FR_MODE: Fast Rotate mode select. G2 Fast Rotate. Transforms a surface via FR_TYPE transformation. Works in either 2-buffer (copy) or 1-buffer mode (in place) mode. The engine breaks down a larger surface into a grid of smaller FR_BLOCKs. Works on the granularity of an FR_BLOCK. An FR_BLOCK is: ▪ 16x16 pixel block (DSTCD = bpp8) ▪ 8x8 pixel block (DSTCD = bpp16) ▪ 4x4 pixel block (DSTCD = bpp32) Maximum surface size is 4096x4096 Key information - source and destination base address must be 128-bit word aligned - engine works on FR_BLOCK granularity: transformed surface width in multiples of 16 bytes** transformed surface height in multiples of 16/8/4 lines for bpp8/bpp16/bpp32 FR_BLOCK if surface dimension is not a multiple, software can program FR engine to transform larger surface (round up to next FR_BLOCK in width and height-->transform-->calculate relative memory pointer address) - during a rotational transformation (TRANS_LR, TRANS_RL, ROT_90, ROT_270): the stride of the output surface != the stride of the input surface when working on non-square input input: output: *^---------- *$---------- $ 0 1 2 3 4 5 6 7 ^ 24 16 8 0 ^ ^ ^ ^ | 8 9 10 11 12 13 14 15 | 25 17 9 1 ^ ^ ^ ^ |16 17 18 19 20 21 22 23 | 26 18 10 2 ^ ^ ^ ^ |24 25 26 27 28 29 30 31 | 27 19 11 3 ^ ^ ^ ^ | - - - - * * * * | 28 20 12 4 * * * * | - - - - * * * * | 29 21 13 5 * * * * | - - - - * * * * | 30 22 14 6 * * * * | - - - - * * * * | 31 23 15 7 * * * * - Tiling alignment restrictions subsume FR restrictions because the FR surface base address is the same as the start address. **Register Programming:** FR_MODE - inplace or copy FR_TYPE - type of transformation DSTCD - bpp8, bpp16, bpp32

Bit	Reset	Description
		SRCBA - source base address SRCWIDTH - (width in pixels-1) SRCHEIGHT - (height in lines-1) SRCS - source stride DSTBA - destination base address DSTS - destination stride FR_READWAIT - always set to enable FR inefficency in the following setup : 1. FR_MODE==SQUARE 2. if(FR_TYPE==YFLIP && SRCHEIGHT==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 Or if(FR_TYPE==XFLIP && SRCWIDTH==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 3. Then, the middle 8 lines/pixels, e.g., YFLIP/16bpp/n==3, line 8 to line 15 will be processed twice. 00 = disable fast rotate - this turns off the second level clock to fr_rotate engine. Please remember to do so! 01 = src/dst copy mode - two separate buffers 10 = square in place - one buffer 11 = blank 0 = DISABLE 1 = SRC_DST_COPY 2 = SQUARE 3 = BLANK
22:21	0x0	CLIPC: Clipping rectangle control, if clip enable, bit 57, XYTDW, should be cleared. 0x=clipping disabled 10=draw only inside clipping rectangle 11=draw only outside clipping rectangle
9	0x0	ALPSRCORDST: 32-bit blending mode, output alpha selection 0:source alpha, 1:destination alpha 0 = DISABLE 1 = ENABLE
8:4	0x0	ALPTYPE: Alpha blending method FIX: ALPHA blending: Fixed alpha, ALPHA is the value, B5G6R5 * B5G6R5 VCAA: B5G6R5->B5G6R5 PL1BPP: ALPHA blending: Alpha 1 bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero. PL2BPP: ALPHA blending: Alpha 2 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero. PL4BPP: ALPHA blending: Alpha 4 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero. PL8BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1 bit from source B5G5R5A1, alpha(MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4 bits from source A4B4G4R4, alpha(LSB). Dest: B5G6R5. PLS4BPP: ALPHA blending: Alpha 4 bits from source B4G4R4A4, alpha(MSB). Dest: B5G6R5. VCAA: R8G8B8A8->B5G6R5 without reading VCAA plane (surface blit with bpp down convert - implemented in hardware by the VCAA engine; not really a VCAA resolve) PLS8BPP: ALPHA blending: Alpha 8 bits from source/destination (decided by ALPSRCORDST), R8G8B8A8, alpha(MSB). DST: R8G8B8A8 VCAA: R8G8B8A8->R8G8B8A8, alpha has same blending method as RGB PLS8BX: ALPHA blending: Alpha 8 bits from source B8G8R8A8, alpha(MSB). Dest: B5G6R5. (**Restrictions) VCAA: R8G8B8A8->B5G6R5 PLS1BPPAL: ALPHA blending: Alpha 1 bit from source A1B5G5R5, alpha(LSB). Dest: B5G6R5 VCAA: A1B5G5R5->A1B5G5R5 **Restriction PLS8BX alpha blending has the following restrictions:

Bit	Reset	Description
		1. Source/destination addresses have to be in a 128-bit boundary. 2. Destination width has to be a multiple of 4 pixels. 3. Source/Destination strides have to be a multiple of 128 bits. 0 = FIX 1 = PL1BPP 2 = PL2BPP 3 = PL4BPP 4 = PL8BPP 5 = PL44BPP 6 = PLS1BPP 7 = PLS4BPPAL 8 = PLS4BPP 9 = PLS8BPP 10 = PLS8BX 11 = PLS1BPPAL
3	0x0	BEWSWAP: Host port word swap 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 0 = DISABLE 1 = ENABLE

14.6.26 G2SB_CTX4_G2CONTROLMAIN_0

G2 Blit Formats:

(1) G2 Blit size is based on height in lines, width in pixels, stride in bytes, and pixel size (color depth). Unless alpha blending is enabled, G2 does not care about pixel components.

Source color depths supported by G2 Blit:

- color depth same as destination: G2CONTROLMAIN.SRCCD = 1
- monochrome: G2CONTROLMAIN.SRCCD = 0

(source monochrome color determined by G2SRCBGC, G2SRCFGC)

Pattern color depths supported by G2 Blit:

- color depth same as destination: G2PATOS.PATCD = 1
- monochrome: G2PATOS.PATCD = 0

(pattern monochrome color determined by G2PATBGC, G2PATFGC)

Destination color depths supported by G2 Blit:

- 1 byte per pixel: G2CONTROLMAIN.DSTCD = BPP8
- 2 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP16
- 4 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP32

(2) Alpha blending operates on 3 or 4 component pixels of a specified color depth. One of the components may be alpha (A), depending on the alpha blend format, which is only allowed in certain component positions defined below. The only format rule of the remaining components is that they must be in the same source and destination position because the alpha blend engine does not support component swapping. For example, valid 32BPP formats are: RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888, GRBA_8888->GRBA_8888, etc.

Source formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16:
  - ALPTYPE_PLS1BPP: xxxA_5551 (for example, BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
  - ALPTYPE_PLS4BPP: xxxA_4444 (for example, BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
  - (other) : xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32:
  - (all) : xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

Destination formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16: xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

src cd	src format	dst cd	dst format	examples
BPP8	no support	BPP8	no support	
BPP16	xyzA_5551, xyzA_4444, xyz_565	BPP16	xyz_565	BGRA_5551->BGR_565, RGB_565->RGB565
BPP32	xyzA_8888	BPP32	xyzA_8888	RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888
BPP32	xyzA_8888	BPP16	xyz_565	RGBA_8888->RGB_565, BGRA_8888->BGR_565

VCAA Resolve Engine

(1) Format support

srcColor format	srcCoverage format	legal dest format(s)
A1B5G5R5	VCAA	A1B5G5R5
R8G8B8A8	VCAA	R8G8B8A8, B5G6R5
B5G6R5	VCAA	B5G6R5

To program resolve type:

- ALPTYPE == FIX: B5G6R5 -> B5G6R5
- ALPTYPE == PLS1BPPAL: A1B5G5R5 -> A1B5G5R5
- ALPTYPE == PLS8BPP: R8G8B8A8 -> R8G8B8A8
- ALPTYPE == PLS8BX: R8G8B8A8 -> B5G6R5

(2) Color surface programming

- color surface base address: SRCBA
- subsurface start x: SRCX (pixel index)
- subsurface start y: SRCY (line index)
- color surface stride: SRCS (bytes)

The engine supports subsurface resolve. One can imagine a full color buffer of 1024x768, but

```
[pseudo_code]
      color_surface_start_address_fetch = pcurctx->regs.rG2SRCBA.uSRCBA() +
       (pcurctx->regs.rG2SRCPS.uSRCX() * VCAAState.color_surface_depth) +
       (pcurctx->regs.rG2SRCPS.uSRCY() * VCAAState.color_surface_stride);
```

(3) VCAA surface programming

- VCAA surface base address: PATBA
- VCAA surface stride: PATST (bytes)

```
[pseudo_code]
      vcaa_surface_start_address_fetch = pcurctx->regs.rG2PATBA.uPATBA();
```

Software *MUST* directly program this register to the proper subsurface location in the VCAA surface which corresponds to the SRCX and SRCY programming. The calculation is PATBA = vcaa_base_address + SRCX * (1) + SRCY * PATST;

(4) Resolve surface programming

- resolve window width: DSTWIDTH (pixels)
- resolve window height: DSTHEIGHT (lines)
- resolve output stride: DSTS (bytes)

These values program the width and height of the resolved surface. In the base case,

- DSTWIDTH = color surface width
- DSTHEIGHT = color sufrace height
- DSTS = resolve surface format bpp * DSTWIDTH

The vcaa engine technically supports DSTX and DSTY (like SRCX and SRCY).

```
  [pseudo_code]
     resolve_surface_start_address_put = pcurctx->regs.rG2DSTBA.uDSTBA() +
       (pcurctx->regs.rG2DSTPS.uDSTX() * resolve_surface_color_depth) +
       (pcurctx->regs.rG2DSTPS.uDSTY() * resolve_surface_color_stride);
```

(5) Resolving a pixel

The coverage surface is a 8 bits per pixel surface (C4X4). The surface is initialized to all 1s by 3d.

For a given pixel, we need to look up 4 bits of coverage data. If the data is 0xf, the pixel does not need to be resolved. Otherwise, we need to reblend the pixel with certain weightings of its neighbors (i.e. we're on the edge of some geometry).

If any coverage bit is 0, we need to resolve the color. To do so, we calculate the following equation:

```
   color_new = (
                 20*color_old +
                 27*(
                     ((cover_down  == 0) ? color_down  : color_old) +
                     ((cover_right == 0) ? color_right : color_old) +
                     ((cover_left  == 0) ? color_left  : color_old) +
                     ((cover_up    == 0) ? color_up    : color_old)
                     )
               ) / 128;
```

If coverage bit == 0, then we use the neighbor color value in the resolve

If coverage bit == 1, then we use the center color value in the resolve

The resolve is done on a per channel (r/g/b/a) basis:

format convert input channel to 8-bit

perform resolve equation above

format convert to output format

For the resolution of the alpha channel:

- In B5G6R5: no alpha information. do nothing.
- In A1B5G5R5: no resolve. keep the source color alpha bit.
- In R8G8B8A8: normal 8-bit channel resolve

(6) Resolving a pixel on the edge

When on an edge, the VCAA engine treats the coverage bit as a 1. Use the center color value. This holds true even on the edge of the subsurface resolve window when located in the middle of the larger surface

(7) Maximum dimensions

The maximum resolve subsurface is 4096 pixels wide.

(8) Surface restrictions

The VCAA engine has the following restrictions with respect to input surfaces

- input base address is 128-bit memory word aligned

- input stride is a multiple of 128 bits (16 bytes)

The VCAA engine has the following restrictions with respect to output surfaces

- output base address is 128-bit memory word aligned

- output stride is a multiple of 128 bits (16 bytes)

(9) Coverage surface layout

The surface is C4X4.

Surface layout (C4X4)

```
(lsb2msb)  C0_C1_C2_C3_X_X_X_X     (X is a don't care.  Qrast inits to 1)
(msb2lsb)  X_X_X_X_C3_C2_C1_C0


     +          o          o
          C0
             C1   Cov(0,1)
        C2
          C3
     o          o          o

       Cov(1,0)   Cov(1,1)


     o          o          o

     +/o = pixel centers
```

In the example above,

```
        Cov(0,0) = X_X_X_X_C3_C2_C1_C0
        C3       = V_C3(Cov(0,0));
        ..         ..
```

```
        C0        = V_C0(Cov(0,0));
    #define V_C3(bits)  (bits & 0x8)
    #define V_C2(bits)  (bits & 0x4)
    #define V_C1(bits)  (bits & 0x2)
    #define V_C0(bits)  (bits & 0x1)
```

Resolving a pixel

In an ideal world, the coverage would be simply laid out such that when resolving Pixel(x,y), we would need only to fetch Cov(x,y) to get all 4 bits of information.

In actuality, the coverage surface was rearranged to help qrast in performance. To resolve Pixel(x,y) gr2d has to fetch 4 C4X4 bytes [Cov(x,y), Cov(x-1,y), Cov(x,y-1), Cov(x-1,y-1)]. From each C4X4 byte, it extracts one bit which is used in the resolve equations shown in (5).

Need the logical vcaa bits (DRLU) for blend which is spread across four C4X4 coverage bytes

```
          U

    L

        o

            R

      D
```

where

o = Pixel(x,y)
D = V_C1(Cov(x-1,y))
R = V_C0(Cov(x,y))
L = V_C3(Cov(x-1,y-1))
U = V_C2(Cov(x,y-1))

Offset: 0x501f | Byte Offset: 0x1407c | Read/Write: R/W | Reset: 0x00000000 (0bx000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: Pattern Data Select: SRCSEL and PATSEL cannot be both enabled. 0 = SCREEN 1 = MEMORY
28	0x0	DSTDIR: Destination direct addressing
27	0x0	SRCDIR: source direct addressing
26:25	0x0	GCSW: Display Switching Window Control (GCSW[1:0]). This parameter controls multi-buffering for Display. x0: At end of current command, do not send signal to Display to switch buffer. 01: two buffers, dstba and dstba_b are used 11: three buffers, dstba, dstba_b, dstba_c are used
24	X	SRCBAS: Source base address select: 0=srcba 1=dstba This is not used for Line Draw and if source data comes from host memory.
23:22	0x0	SRCT: Source transparency enable: 0x=source transparency disabled, 10=mono source background transparency or color source transparency,

Bit	Reset	Description
		11=mono source foreground transparency or inverse color source transparency. NOTE: When source transparency is enabled and SRCCD==0(mono) SRCBGC!=SRCFGC should be satisfied.
21	X	HLMONO: Start from msb or lsb in byte when mono expansion If HLMONO is 1, bit 0 (the lsb) is the first bit. If HLMONO is 0, bit 7 (the msb) is the first bit
20	X	SRCCD: 0: Source mono 1: Source has same color depth as destination SRCCD==0 xdir/ydir has to be 0
19:18	0x0	DSTT: Destination read transparency enable: 0x=destination read transparency disabled 10=color destination read transparency 11=inverse color destination read transparency. 20 rw INIC Initiate Command (1=initiate command immediately, 0=wait for launch write)
17:16	X	DSTCD: Destination color depth: 00=8-bpp 01=16-bpp 10=32-bpp 11=reserved 0 = BPP8 1 = BPP16 2 = BPP32 3 = RESERVED3
15	0x0	SRCSEL: Source Data Select: 0 = SCREEN 1 = MEMORY
14	0x0	YFLIP: flip y direction to make image upside down or the other way. If YFLIP==1, ROP cannot include destination. 0 = DSIABLE 1 = ENABLE
13	0x0	PATPACK: Pattern Data is in Pack Mode. PATLNGAP in G2PATPACK is the line gap for pattern packed mode If (PATPACK && ~PATSEl), pattern data is packed and from screen, PATMONOW/PATMONOH should be programmed properly to fetch pattern data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
12	0x0	SRCPACK: Source Data is in Pack Mode. SRCLNGAP in G2SRCPACK is the line gap for source packed mode. If (SRCPACK && ~SRCSEL), source data is packed and from screen, SRCMONOW/SRCMONOH should be programmed properly to fetch data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
11	0x0	XYTDW: xy transpose, Line stride DSTS has to be 16-byte aligned if enabled. If XYTDW==1, ROP cannot include destination.
10	0x0	YDIR: 0=incrementing, 1=decrementing. ydir should not be set when source surface has different color depth as destination surface. SRCCD==0 (mono src) OR PATCD==0 (mono pat), ydir has to be 0
9	0x0	XDIR: 0=incrementing, 1=decrementing. xdir should not be set when source surface has different color depth as destination surface. e.g. ALPTYPE=PLS8BX (32bpp blending with 16bpp), xdir has to be 0 SRCCD==0 (mono src) OR PATCD==0 (mono pat), xdir has to be 0
8	0x0	PATFL: When mono pattern is set, we use mono tile pattern fill. Current pattern client can

Bit	Reset	Description

		support a 16x16 bit tile which can be used to generate a larger surface (i.e., tile this surface in the x and y direction) The tile is stored in memory.
		Currently, PATXO and PATYO indicate where to start in the 16x16 tile when expanding the monochrome data. See patxo and patyo comments below for programming. When xdir==1 Patxo = pattern width - (patxo+destination width)&0xF -- For xoffsets y offsets PATYO remains the same when xdir/ydir change. How the tile replication pattern looks based on xdir/ydir (without programming patxo above): xdir=0 ydir=0 xdir=1 ydir=0 +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ // xdir=0 ydir=1 xdir=1 ydir=1 Mono tile is +---+ |^|^| |~~~| +---+
		0 = DISABLE
		1 = ENABLE

7	0x0	PATSLD: BitBlt Solid Pattern Fill: 1=enable. BGC will be used as the color value. 0 = DISABLE 1 = ENABLE
6	0x0	SRCSLD: BitBlt Solid Source Color Fill: 1=enable. FGC will be used as the color value. 0 = DISABLE 1 = ENABLE
5	0x0	ALPEN: BitBlt Alpha Blending, 1=enable. 0=disable, when both Faden and alpen are 1, output=Source*alpha_v + fadoff, alpha_v is decided by alptype
4	0x0	FADEN: BitBlt Source Copy Fade enable, 1=enable (share with mltln), only support 16bpp mode 0 = DISABLE 1 = ENABLE
3	0x0	TEST0BIT: Command finish timing bit 0: 2D command finishes when last data has been pushed to memory write client. 1: 2D command waits memory write client to be idle to finish. 0 = DISABLE 1 = ENABLE
2	0x0	TURBOFILL: fast fill rectangle in 128 bits/clock. Some limitations with this mode:srcsld==1 rop==0xcc, no clipping, no transparency, xdir==0, ydir==0, flip==0, xytdw==0. Results are undefined if the above limitations are not satisfied.
1:0	0x0	CMDT: Command Type: 00=BitBlt 01=Line Draw 10=VCAA 11=reserved When the raise command is executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

14.6.27 G2SB_CTX4_G2ROPFADE_0

Offset: 0x5020 │ Byte Offset: 0x14080 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:24	X	FADOFF:
23:16	X	FADCOE:
7:0	0x0	ROP: If YFLIP==1 or XYTDW==1, ROP cannot include destination. Since destination may have been corrupted before reading out.

14.6.28 G2SB_CTX4_G2ALPHABLEND_0

Offset: 0x5021 | Byte Offset: 0x14084 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
31:24	X	ALPHA1V
23:16	X	ALPHA0V
8	0x0	ALPHAINV
7:0	X	ALPHA

14.6.29 G2SB_CTX4_G2CLIPLEFTTOP_0

ClipRect Methods. Top-left bounds are inclusive

Offset: 0x5022 | Byte Offset: 0x14088 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPT
14:0	X	CLIPL

14.6.30 G2SB_CTX4_G2CLIPRIGHTBOT_0

Bottom-right bounds are exclusive

Offset: 0x5023 | Byte Offset: 0x1408c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPB
14:0	X	CLIPR

14.6.31 G2SB_CTX4_G2PATPACK_0

Pattern Packed Mode

Pattern methods: G2PATPACK should be used to specify the line gap. Use G2PATPACK_SIZE to program height and width. PACK is generally only useful with narrow monochrome surfaces

Note: PACK is not officially supported when the surface is tiled.

Offset: 0x5024 | Byte Offset: 0x14090 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

14.6.32 G2SB_CTX4_G2PATPACK_SIZE_0

Pattern Packed Mode

Packed mode, pattern data line gap. Byte

G2PATPACK_SIZE -- extension of G2PATPACK, this register only holds the size of the packed data

> **Note:** PACK is not officially supported when the surface is tiled.

Offset: 0x5025 | Byte Offset: 0x14094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	PATMONOH: Pattern mono data height in packed mode, byte
15:0	X	PATMONOW: Mono data width in packed mode, byte

14.6.33 G2SB_CTX4_G2PATBA_0

Offset: 0x5026 | Byte Offset: 0x14098 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBA: If (PATFL==1){ It has to be 16-byte aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

14.6.34 G2SB_CTX4_G2PATOS_0

Offset: 0x5027 | Byte Offset: 0x1409c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00xxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	PATYO: y offset for mono tile pattern fill. see PATFL
27:24	X	PATXO: x offset for mono tile pattern fill. see PATFL
22:21	0x0	PATT: Mono pattern transparency enable: 0x=pattern transparency disabled, 10=mono pattern background transparency or color pattern transparency, 11=mono pattern foreground transparency or inverse color pattern transparency. NOTE: When pattern transparency is enabled and PATCD==0(mono) PATBGC!=PATFGC should be satisfied.
16	X	PATCD:0 mono1 same as dstcd PATCD==0 xdir/ydir has to be 0
15:0	X	PATST: stride

14.6.35 G2SB_CTX4_G2PATBGC_0

Offset: 0x5028 | Byte Offset: 0x140a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

14.6.36 G2SB_CTX4_G2PATFGC_0

Offset: 0x5029 | Byte Offset: 0x140a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

14.6.37 G2SB_CTX4_G2PATKEY_0

Offset: 0x502a | Byte Offset: 0x140a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

14.6.38 G2SB_CTX4_G2DSTBA_0

Offset: 0x502b | Byte Offset: 0x140ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA: Destination base address (byte address)

14.6.39 G2SB_CTX4_G2DSTBA_B_0

Offset: 0x502c | Byte Offset: 0x140b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_B: Destination base address (byte address) only usable in hardware trigger mode by enable gcsw

14.6.40 G2SB_CTX4_G2DSTBA_C_0

Offset: 0x502d | Byte Offset: 0x140b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_C: Destination base address (byte address)

14.6.41 G2SB_CTX4_G2DSTST_0

Offset: 0x502e | Byte Offset: 0x140b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

14.6.42 G2SB_CTX4_G2SRCPACK_0

Source Data Packed Mode

Surface methods: G2SRCPACK should only be used to specify the line gap. Use G2SRCPACK_SIZE to program height and width

Offset: 0x502f | Byte Offset: 0x140bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

14.6.43 G2SB_CTX4_G2SRCPACK_SIZE_0

Source Data Packed Mode

In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, the required width/stride is 16 byte aligned. Packed mode - source mono data line gap

G2SRCPACKS_SIZE -- extension of G2SRCPACK, this register only holds the size of the packed data

Offset: 0x5030 | Byte Offset: 0x140c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCMONOH: Packed mode - source mono data height in bytes
15:0	X	SRCMONOW: Packed mode - source mono data width in bytes

14.6.44 G2SB_CTX4_G2SRCBA_0

Source Base Address

Offset: 0x5031 | Byte Offset: 0x140c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA: Source base address (byte address)

14.6.45 G2SB_CTX4_G2SRCBA_B_0

SB only

Offset: 0x5032 | Byte Offset: 0x140c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA_B: This parameter specifies the start address of source image stored in the image buffer memory. In 4:2:0 format mode, this image block accommodates for Y-image. This address specifies byte position, however, bits [2:0] are restricted with respect to the data formats to fit multiple pixels in one memory word (8 bytes). For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format. Since one Y pixel takes 8-bit, all 8 byte positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there are no restrictions on this value.)

14.6.46 G2SB_CTX4_G2SRCST_0

Offset: 0x5033 | Byte Offset: 0x140cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS: Source Stride coordinate (bytes) with respect to SRCBA. In order to fit multiple pixels in one memory word (8 bytes), bits [2:0] are restricted with respect to the data formats. For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format.

14.6.47 G2SB_CTX4_G2SRCBGC_0

srcColors

Offset: 0x5034 | Byte Offset: 0x140d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

14.6.48 G2SB_CTX4_G2SRCFGC_0

Offset: 0x5035 | Byte Offset: 0x140d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

14.6.49 G2SB_CTX4_G2SRCKEY_0

Offset: 0x5036 | Byte Offset: 0x140d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

14.6.50 G2SB_CTX4_G2SRCSIZE_0

Offset: 0x5037 | Byte Offset: 0x140dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	SRCHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	SRCWIDTH

14.6.51 G2SB_CTX4_G2DSTSIZE_0

Offset: 0x5038 | Byte Offset: 0x140e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	DSTHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	DSTWIDTH: In 2D mode, the largest number can be programmed is 0x7fef

14.6.52 G2SB_CTX4_G2SRCPS_0

ImageBlit Methods

Offset: 0x5039 | Byte Offset: 0x140e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCY
15:0	X	SRCX:SRCX[2:0] are ignored in SRCCD==0 (mono expansion). The first bit of the first byte (bit7 if HLMONO==0, or bit0 if HLMONO==1) always expand to DSTX,DSTY

14.6.53 G2SB_CTX4_G2DSTPS_0

Offset: 0x503a | Byte Offset: 0x140e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	DSTY
15:0	X	DSTX: NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero

14.6.54 G2SB_CTX4_G2CBDES_0

Circular Buffer

Offset: 0x503b | Byte Offset: 0x140ec | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TOPCLIP :top clipping at the first buffer (buffer start), refer to TOP_CBLINE in G2CBDES2 0 = DISABLE 1 = ENABLE
30:16	X	CBLINE: vertical line number in one buffer
7:0	X	CBCOUNT: This specifies the number of buffers in circular buffer feature

14.6.55 G2SB_CTX4_G2CBSTRIDE_0

Offset: 0x503c | Byte Offset: 0x140f0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	X	CBUVSTRIDE: Chroma Buffer Stride default is half of luma 00= 1/2 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 2 bytes. 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 4 bytes. 1x= Reserved 0 = CBS2X 1 = CBS1X 2 = CBS4X
23:0	X	CBSTRIDE: Video Buffer Luma(or RGB) Buffer Stride. This is luma buffer stride (in bytes)

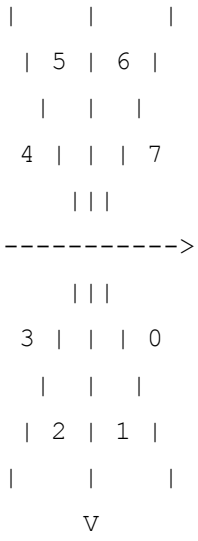
14.6.56 G2SB_CTX4_G2LINESETTING_0

Line Methods

Offset: 0x503d | Byte Offset: 0x140f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	OCTANTS: 000: octant 0 001: octant 1 010: octant 2 011: octant 3 100: octant 4 101: octant 5 110: octant 6 111: octant 7
28	X	LINEUSEOCTANT: Use OCTANTS in G2LINEDELTAN register instead of MAJOR LINEXDIR LINEYDIR
27	X	DROPLASTP: Draw last pixel or not
26	X	LINEYDIR
25	X	LINEXDIR
24	X	MAJOR: 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

14.6.57 G2SB_CTX4_G2LINEDELTAN_0



Offset: 0x503e | Byte Offset: 0x140f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

14.6.58 G2SB_CTX4_G2LINEDELTAM_0

Offset: 0x503f | Byte Offset: 0x140fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

14.6.59 G2SB_CTX4_G2LINEPOS_0

Offset: 0x5040 | Byte Offset: 0x14100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LINEYPOS
15:0	X	LINEXPOS

14.6.60 G2SB_CTX4_G2LINELEN_0

Offset: 0x5041 | Byte Offset: 0x14104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

14.6.61 G2SB_CTX4_G2CSCFOURTH_0

Offset: 0x5042 | Byte Offset: 0x14108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:16	X	G2V: Multiplier for G for V generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.368 (decimal) or 0x12F. For any other combination, this parameter is ignored.
8:0	X	G2U: Multiplier for G for U generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.291 (decimal) or 0x125. For any other combination, this parameter is ignored.

14.6.62 G2SB_CTX4_G2SRCST_B_0

Offset: 0x5043 | Byte Offset: 0x1410c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

14.6.63 G2SB_CTX4_G2UVSTRIDE_0

Offset: 0x5044 | Byte Offset: 0x14110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

14.6.64 G2SB_CTX4_G2CBDES2_0

Circular Buffer Controller 2

Offset: 0x5045 | Byte Offset: 0x14114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

14.6.65 G2SB_CTX4_G2TILEMODE_0

Offset: 0x5046 | Byte Offset: 0x14118 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx0xxx0xxx0xxx0xxx0xxx0)

Bit	Reset	Description
20	0x0	DST_WR_TILE_MODE: destination surface 0 = LINEAR 1 = TILED
16	0x0	DST_RD_TILE_MODE: Same as destination write unless DST_RD_WR_SEP (not supported) 0 = LINEAR 1 = TILED
12	0x0	PAT_UV_TILE_MODE:UNUSED 0 = LINEAR 1 = TILED
8	0x0	PAT_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED
4	0x0	SRC_UV_TILE_MODE:UV surface, ignored in RGB mode 0 = LINEAR 1 = TILED
0	0x0	SRC_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED

14.6.66 G2SB_CTX4_G2PATBASE_0

Offset: 0x5047 | Byte Offset: 0x1411c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PAT_BASE: pattern base address in tile mode, PATBA is the linear address where pixel start

14.6.67 G2SB_CTX4_G2SRCBA_SB_SURFBASE_0

SB_SURFBASE Registers

These registers need only be programmed when using SB. They point to the base of the various source and destination surfaces. Technically they are only needed when tiling is enabled, but there is no harm in always programming them.

Their counterpart registers (_SB_SURFBASE with SB_SURFBASE stripped off) indicate the location of the first pixel to be sourced or written within the surface.

This register exists to mimic the X, Y (SRCX, SRCY) functionality found in the BitBlt engine.

For example, to get at pixel X, Y:

SRCBA = SRCBA_SB_SURFBASE + Y*stride + X*Bpp

Offset: 0x5048 | Byte Offset: 0x14120 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRC_ADDR: surface address corresponding to G2SRCBA: -base of interleaved sources (RGB, YUV) -base of Y plane. Only used by the StretchBlit Engine

14.6.68 G2SB_CTX4_G2DSTBA_SB_SURFBASE_0

Offset: 0x5049 | Byte Offset: 0x14124 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_ADDR: Surface address corresponding to G2DSTBA Only used by the StretchBlit Engine

14.6.69 G2SB_CTX4_G2DSTBA_B_SB_SURFBASE_0

Offset: 0x504a | Byte Offset: 0x14128 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_B_ADDR: Surface address corresponding to G2DSTBA_B. Only used by the StretchBlit Engine, and G2CONTROLSB.DBSEL() is enabled

14.6.70 G2SB_CTX4_G2VBA_A_SB_SURFBASE_0

Offset: 0x504b | Byte Offset: 0x1412c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	V_ADDR: Surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of V plane. Only used by the StretchBlit Engine

14.6.71 G2SB_CTX4_G2UBA_A_SB_SURFBASE_0

Offset: 0x504c | Byte Offset: 0x14130 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	U_ADDR: Surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of U plane. Only used by the StretchBlit Engine

14.7 G2SB CTX 5 Registers

14.7.1 G2SB_CTX5_INCR_SYNCPT_0

Memory Controller Tiling Definitions

Offset: 0x2000 | Byte Offset: 0x8000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIAT6 1 = OP_DONE 2 = RD_DONE

Bit	Reset	Description
		3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	INDX: syncpt index value

14.7.2 G2SB_CTX5_INCR_SYNCPT_CNTRL_0

Offset: 0x2001 │ Byte Offset: 0x8004 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

14.7.3 G2SB_CTX5_INCR_SYNCPT_ERROR_0

Offset: 0x2002 │ Byte Offset: 0x8008 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

14.7.4 G2SB_CTX5_G2CLASSCHANNEL_REGONLY_0

This register can be accessed by both register read/write and command FIFO in order not to break existing code , keep suffix REGONLY.

2D syncpt behavior

- Each 2D context has its own syncpt client.
- OP_DONE increments when 2D's memory write client is idle, meaning all preceding operations are complete.
- RD_DONE behaves like OP_DONE
- REG_WR_SAFE behaves like IMMEDIATE meaning a syncpt increment returns immediately and does not indicate that a register write will NOT block. The implications are that 2D's host FIFO can fill and block access to 2D, which will indeterminately block a direct CPU access.

Offset: 0x2008 │ Byte Offset: 0x8020 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID

Bit	R/W	Reset	Description
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

14.7.5 G2SB_CTX5_G2TRIGGER_0

Control methods, since they are not programmed often, put them in the beginning

Offset: 0x2009 | Byte Offset: 0x8024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

14.7.6 G2SB_CTX5_G2TRIGGER1_0

Offset: 0x200a | Byte Offset: 0x8028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

14.7.7 G2SB_CTX5_G2TRIGGER2_0

Offset: 0x200b | Byte Offset: 0x802c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

14.7.8 G2SB_CTX5_G2CMDSEL_0

Offset: 0x200c | Byte Offset: 0x8030 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxx0xxxxxxxx00x0x0000000xxxx)

Bit	R/W	Reset	Description
28	RW	0x0	PRIORITY: indicates priority of the context, note vi-triggered contexts take priority over HIGH 0 = LOW 1 = HIGH
27:25	RW	X	LINKVAL
24	RW	0x0	LINKEN: Test purpose
23:16	RW	X	BUFFER_INDEX
15	RW	0x0	FRAME_END
14	RW	0x0	FRAME_START: host writes this bit to tell which buffer is ready
10	RW	0x0	CLIP_SOURCE_TOP_BOTTOM: Reserved for hardware test. Software should not use this bit. 0 = DISABLE 1 = ENABLE
9:8	RW	0x0	G2OUTPUT: 00: to memory 01: to EPP 0 = MEMORY 1 = EPP 2 = RESERVED2 3 = RESERVED3

Bit	R/W	Reset	Description
7	RW	0x0	CBSBDISABLE
6	RW	0x0	HOSTTRIGGER: host trigger the command, host also need program circular buffer related data
5	RW	0x0	VITRIGGER:VI trigger enable, can be circular buffer or full frame
4	RW	0x0	CBENABLE: circular buffer feature enable 0= disable 0 = DISABLE 1 = ENABLE
0	RO	X	SBOR2D: 0 = G2 1 = SB

14.7.9 G2SB_CTX5_G2RAISE_0

Offset: 0x200d │ Byte Offset: 0x8034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE: Push back to read FIFO when all commands in the channel are done.

14.7.10 G2SB_CTX5_G2HOSTSET_0

Offset: 0x200f │ Byte Offset: 0x803c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	HSTFPXL: specifies the first pixel location (in byte) in the first source data entry. In mono mode pixel should always start in byte boundary
3:0	0x0	HSTLNGAP: specifies distance (in byte) from last pixel of a line to the first pixel of next line. In mono mode, the value is the nearest byte number. e.g., if gap is 3 bits, HSTLNGAP should be 1 (byte), if gap is 9 bits, HSTLNGAP should be 2 (byte)

14.7.11 G2SB_CTX5_G2HOSTFIFO_0

Offset: 0x2010 │ Byte Offset: 0x8040 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

14.7.12 G2SB_CTX5_G2VDDA_0

Offset: 0x2011 │ Byte Offset: 0x8044 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VDSTEP: Vertical Scaling DDA. Reset value: xxxx-xxxxh. Vertical DDA Step (Increment) Value (VDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for vertical scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_height-1-VDTINI) / (Actual_destination_height-1). Truncate the remaining bits to keep the 12-bit fraction. Since we have to meet (Actual_source_height-1)*1.0 >= (Actual_destination_height - 1)*VDSTEP + VDTINI. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half line-spacing for up to approximately 2000

Bit	Reset	Description
		target lines. For example, image expansion from 240 lines to 400 lines VDSTEP[17:0] = 19'b00_0000_1001_1001_1010 and image contraction from 240 lines to 150 lines VDSTEP[17:0] = 19'b00_0001_1001_1001_1010.

14.7.13 G2SB_CTX5_G2VDDAINI_0

Offset: 0x2012 | Byte Offset: 0x8048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	VDBINI: Vertical DDA Initial Value for Bottom Field (VDBINI[7:0]) (see description above)
7:0	X	VDTINI: Vertical Scaling DDA Initial Values. Vertical DDA Initial Value for Top-Field (VDTINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for vertical scaling. Given a pair of consecutive source image lines representing positions 0.0 and 1.0, a destination line-image will be created at any position in [0.0, 1.0), that is specified by the fraction part of the vertical DDA. The 8-bit initial fraction value specifies the position of the first output (destination) line-image located between the first and the second input (source) line-images, respectively at position 0.0 and 1.0. This parameter provides a way to compensate relative displacement of top and bottom fields of a source image. Suppose, 2-to-1 interlaced original video field images #1 and #2 are in the image buffer, to be displayed in an overlay window. Assume that the #1 field image is the top-field. Using VDTINI = 8'hC0 for the top-field image and VDBINI = 8'h40 for the bottom-field image, for example, the two images from #1 and #2 fields get mapped to identically positioned destination images in the overlay window.

14.7.14 G2SB_CTX5_G2HDDA_0

Offset: 0x2013 | Byte Offset: 0x804c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HDSTEP: Horizontal Scaling DDA. Reset value: xxxx-xxxxh. Horizontal DDA Step (Increment) Value (HDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for horizontal scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_width-1-HDINI) / (Actual_destination_width-1). Truncate the remaining bits to keep the 12-bit fraction. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half pixel-spacing for up to approximately 2000 destination pixels. For example, image expansion from 720 pixels to 800 pixels HDSTEP[17:0] = 19'b000_0000_1110_0110_0111 and image contraction from 720 pixels to 250 pixels HDSTEP[17:0] = 19'b000_0010_1110_0001_0101.

14.7.15 G2SB_CTX5_G2HDDAINILS_0

Offset: 0x2014 | Byte Offset: 0x8050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	HDINI: Horizontal Scaling DDA Initial Value. Horizontal DDA Initial Value (HDINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for horizontal scaling. Given a group of six consecutive source pixels that the two pixels at the center representing positions 0.0 and 1.0, a destination pixel will be created at any position in [0.0, 1.0), that is specified by the fraction part of the horizontal DDA. The 8-bit initial fraction value specifies the position of the first output (destination) pixel located between the first and the second input (source) pixels, respectively at position 0.0 and 1.0.For

Bit	Reset	Description
		horizontal scaling this value may normally be set to 0.

14.7.16 G2SB_CTX5_G2CSCFIRST_0

The next 3 registers contain 8 parameters used by the YCbCr (YUV) to RGB color space conversion logic. This logic can also be used for RGB gain/gamma correction in RGB to RGB modes, and for luma gain in YUV to YUV modes.

Two new parameters (G2U, G2V) have been added for RGB to YUV conversion. These are in address 0x38. The equations have the form:

if (yuv2rgb)

(r,g,b) = matrix*(y+sbreg_yos,u,v)

matrix: sbreg_cyx, sbreg_cur, sbreg_cvr,
sbreg_cyx, sbreg_cug, sbreg_cvg,
sbreg_cyx, sbreg_cub, sbreg_cvb,

else if (rgb2rgb)

(r,g,b) = matrix*(r,g+sbreg_yos,b)

matrix: sbreg_cvr, 0, sbreg_cur,
sbreg_cvg, sbreg_cyx, sbreg_cug,
sbreg_cvb, 0, sbreg_cub,

else if (yuv2yuv)

(y,u,v) = matrix*(y,u,v)

matrix: sbreg_cub, 0, sbreg_cvb,
sbreg_cug, sbreg_cyx, sbreg_cvg,
sbreg_cur, 0, sbreg_cvr,

else if (rgb2yuv)

(y,u,v) = matrix*(r,g,b) + (sbreg_yos,0,0)

matrix: sbreg_cvb, sbreg_cyx, sbreg_cub,
sbreg_cvg, sbreg_g2u, sbreg_cug,
sbreg_cvr, sbreg_g2v, sbreg_cur,

Offset: 0x2015 │ Byte Offset: 0x8054 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	YOS:Y-Offset (YOFFSET[7:0]) for YUV generation. This parameter consists of 8-bit 2's complement in the range [-128,127]. For YUV->RGB, the recommended value is -16 (decimal) or 0xF0. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV the recommended value is +16 (decimal) or 0x10
21:12	X	CVR: multiplier for V/R for V or R generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 1.5960 (decimal) or 0x0CCIf source data is in RGB format, this parameter may be used as gain adjustment for R component. For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038.
9:0	X	CUB: multiplier for U/B for Y or B generation. Consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 2.0172 (decimal) or 0x102. If source data is in RGB format, this parameter may be used as gain adjustment for B component. This register changes precision when doing RGB to YUV conversion (SIFMT=1xxx, DIFMT=0xxx). CUB becomes s1.8 and the recommended value is +0.098 or 0x019.

14.7.17 G2SB_CTX5_G2CSCSECOND_0

Offset: 0x2016 | Byte Offset: 0x8058 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	CYX: multiplier for Y/G (G gain). This positive-only parameter consists of 8-bit magnitude (1.7). For YUV->YUV, the recommended value is 1.1644 (decimal) or 0x95. For YUV->YUV or RGB->RGB, this may be used as gain adjustment for Y or G component. For RGB->YUV, the recommended value is +0.504 (decimal) or 0x041
21:12	X	CUR: multiplier for U/B for V or R generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This parameter takes non-zero value if hue is rotated. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.071 (decimal) or 0x209
8:0	X	CUG: multiplier for U/B for U or G generation. Consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB, the recommended value is -0.3918 (decimal) or 0x132 For RGB->RGB, this parameter should be set to 0. For YUV->YUV, this parameter should be set to 1 (0x080). For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038

14.7.18 G2SB_CTX5_G2CSCTHIRD_0

Offset: 0x2017 | Byte Offset: 0x805c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	CVB: multiplier for V/R for Y or B generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This coefficient takes non-zero value if hue is rotated. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is +0.257 (decimal) or 0x021
8:0	X	CVG: multiplier for V/R for U or G generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB, the recommended value is -0.8130 (decimal) or 0x168. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.148 (decimal) or 0x113

14.7.19 G2SB_CTX5_G2CMKEYL_0

When key generation is enabled, the value of this register is the lower color/chroma limit and G2CMKEYU is the upper color/chroma limit for component pixel data. Three component signal values (YCrCb/YUV or RGB) of every output pixel are compared to a set of three ranges that are specified by three pairs of lower and upper color/chroma key values.

If the Key Polarity is 0, the Key is set to 1 only when all three component values of an input pixel are respectively within the set of ranges (inclusive of the limit values); otherwise the Key is set to 0.

If the Key Polarity is 1, the Key is set to 1 when any of the three component values of an input pixel are outside the set of ranges; otherwise the Key is set to 0.

StretchBLT, Color/Chroma Key Lower Limit Register

Offset: 0x2018 | Byte Offset: 0x8060 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRL: R or Cr Color Chroma Key Lower Limit (CKRL[7:0]). Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGL:G or Cb Color Chroma Key Lower Limit (CKGL[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.

Bit	Reset	Description
7:0	X	CKBL: B or Y Color/Chroma Key Lower Limit (CKBL[7:0])

14.7.20 G2SB_CTX5_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0x2019 | Byte Offset: 0x8064 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRU: R or Cr Color Chroma Key Upper Limit (CKRU[7:0]) Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGU: G or Cb Color/Chroma Key Upper Limit (CKGU[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity the minimum signal level to the maximum signal level.
7:0	X	CKBU: This is B or Y color/chroma key upper limit value.

14.7.21 G2SB_CTX5_G2UBA_A_0

StretchBLT, Source Image Data U base address A-buffer

Offset: 0x201a | Byte Offset: 0x8068 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SU1SA: Start Address of Source U-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.7.22 G2SB_CTX5_G2VBA_A_0

Offset: 0x201b | Byte Offset: 0x806c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SV1SA: Start Address of Source V-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.7.23 G2SB_CTX5_G2SBFORMAT_0

Offset: 0x201c | Byte Offset: 0x8070 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12:8	X	DIFMT: Destination Image Data Format This parameter defines the data format of destination output. There are two groups of data formats, RGB and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement. 00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement

Bit	Reset	Description
		00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 01000: bpp16 5-6-5 {R[4:0], G[5:0], B[4:0]} 01001: RESERVED 01010: RESERVED 01011: RESERVED 01100: bpp16 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]} 01101: RESERVED 01110: R8G8B8A8 01111: B8G8R8A8 1xxxx: RESERVED 0 = U8Y8V8Y8_OB 1 = Y8U8Y8V8_OB 2 = Y8V8Y8U8_OB 3 = V8Y8U8Y8_OB 4 = U8Y8V8Y8_TC 5 = Y8U8Y8V8_TC 6 = Y8V8Y8U8_TC 7 = V8Y8U8Y8_TC 8 = B5G6R5 9 = RESERVED9 10 = RESERVED10 11 = RESERVED11 12 = B5G6R5BS 13 = RESERVED13 14 = R8G8B8A8 15 = B8G8R8A8 16 = RESERVED16 17 = RESERVED17 18 = RESERVED18 19 = RESERVED19 20 = RESERVED20 21 = RESERVED21 22 = RESERVED22 23 = RESERVED23 24 = RESERVED24 25 = RESERVED25 26 = RESERVED26 27 = RESERVED27 28 = RESERVED28 29 = RESERVED29 30 = RESERVED30 31 = RESERVED31
4:0	X	SIFMT: This parameter defines the data format of source input. There are two groups of data formats, RGB format and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement. 00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 01000: B5G6R5 5-6-5 {R[4:0], G[5:0], B[4:0]} 01001: RESERVED 01010: RESERVED 01011: RESERVED 01100: B5G6R5 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]}

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>
01101: RESERVED
01110: R8G8B8A8
01111: B8G8R8A8
1xxxx: RESERVED
StretchBlit Inputs: RGB inputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV420 input = {YUV420 is converted into 4:2:2 UYVY via memory client} YUV422 inputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}
StretchBlit Outputs: RGB outputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV422 outputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}
StretchBlit Input/Output Rules
<table>
<tr><th>src format</th><th>internal sb format</th><th>dst format</th></tr>
<tr><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td><td>R8G8B8A8</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
</table>
StretchBlit color space converter supports YUV->RGB, YUV->YUV (gain) and RGB->RGB (gain). There is no support for RGB->YUV.
0 = U8Y8V8Y8_OB
1 = Y8U8Y8V8_OB
2 = Y8V8Y8U8_OB
3 = V8Y8U8Y8_OB
4 = U8Y8V8Y8_TC
5 = Y8U8Y8V8_TC
6 = Y8V8Y8U8_TC
7 = V8Y8U8Y8_TC
8 = B5G6R5
9 = RESERVED9
10 = RESERVED10
11 = RESERVED11
12 = B5G6R5BS
13 = RESERVED13
14 = R8G8B8A8
15 = B8G8R8A8
16 = RESERVED16
17 = RESERVED17
18 = RESERVED18
19 = RESERVED19
20 = RESERVED20
21 = RESERVED21
22 = RESERVED22
23 = RESERVED23
24 = RESERVED24
25 = RESERVED25
26 = RESERVED26
27 = RESERVED27
28 = RESERVED28
29 = RESERVED29
30 = RESERVED30
31 = RESERVED31
</td></tr>
</table>

14.7.24 G2SB_CTX5_G2CONTROLSB_0

Offset: 0x201d │ Byte Offset: 0x8074 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000xxx)

Bit	Reset	Description
31	0x0	DISDW: Output destination writes go either to image memory or the EPP. 0: Output data is sent to memory 1: YUV or RGB data is passed directly to the EPP module, and no destination writes take place. 0 = GOTO_IMAGE_BUFFER 1 = GOTO_EPP
30	0x0	ENDITH: Enable Dithering (ENDITH). For 16 bit RGB output modes, the LSB of the color components can be modified by adding a variable residual value that will reduce the banding artifacts that can appear on the display. 0: Normal operation 1: Enable Dithering 0 = DISABLE 1 = ENABLE
28	0x0	KPOL: Key Signal Polarity (KPOL). Color/Chroma key signal is generated by comparing source input pixel color to a range of a color specified by lower and upper limit values. The key signal is interpreted in two ways, depending on which one of video and graphics images is foreground (and the other is background). This is effective only if Key signal generator is enabled. (see G2CMKEYL, G2CMKEYU) 0: Key signal is set to 1 when source pixel is within the lower and upper limit color range. 1: Key signal is set to 1 when source pixel is outside the lower and upper limit color range. 0 = WITHIN_BOUNDS 1 = OUTSIDE_BOUNDS
27	0x0	KEYEN: Key Signal Generator Enable (KEYEN). Key signal generator generates either chroma key signal (from YCbCr signal) or color key signal (from RGB signal) 0: Key signal generator is disabled. 1: Key signal generator is enabled. 0 = DISABLE 1 = ENABLE
26	0x0	DBSEL: StretchBLT Destination Buffer Selection (DBSEL) StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the destination video area. This bit selects one of the two buffering blocks to which this StretchBLT command delivers the destination image. The two buffering memory blocks are called A and B. 0: Destination image goes to ‘A’ buffer. 1: Destination image goes to ‘B’ buffer. 0 = DST_A 1 = DST_B
25	0x0	SBSEL: StretchBLT Source Buffer Selection (SBSEL) StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the source video area. This bit selects one of the two buffering blocks from which this StretchBLT command receives the source image. The two buffering memory blocks are called source-A and source-B. 0: Source image comes from ‘source-A’ buffer. 1: Source image comes from ‘source-B’ buffer. 0 = SRC_A 1 = SRC_B
24	0x0	SITYPE: StretchBLT Source Type (SITYPE). This bit identifies two types of source images. If source image is 2-to-1 interlaced and StretchBLT processes either one of the two interlaced

Bit	Reset	Description
		field-images, physical (positional) displacement between the two interlaced fields must be taken into account. One field-image that is placed higher in position than the other field-image is called top-field and the other is called bottom-field. StretchBLT processing has to lower the top-field (or raise the bottom-field) to match the corresponding two target images in the overlay window (that is progressively scanned) right at the same position. If source image is full frame-image obtained from two interlaced field-images, its type is 'top-field'. If source images are progressively scanned, the type identification is not significant and they may be designated either one of the two types 0: Source image is 'top-field'. 1: Source image is 'bottom-field'. 0 = TOP_FIELD 1 = BOTTOM_FIELD
23	0x0	RANGEREDFRM:In the VC-1 specification, when the value of RANGEREDFRM variable (1-bit) for a picture is equal to 1, the picture shall be scaled up according to the following equation:Y = clip(((Y-128)*2) + 128);Cb = clip(((Cb-128)*2) + 128);Cr = clip(((Cr-128)*2) + 128);The clip operator limits the output to [0, 255].The input is also limited to [0, 255]. In the VC-1 nomenclature, the output (Y, Cb, Cr) tuple corresponds to the 'decoded' picture. The input (Y, Cb, Cr) tuple corresponds to the 'reconstructed' picture. The above equations create the decoded picture while keeping the reconstructed picture intact.Only YUV _OB formats are supported with range reducation enabled, not the YUV _TC or RGB formats The YUV data coming from the video frame buffers (reconstructed picture) in the SDRAM was scaled up using the above equations and given to the display. the value of 1-bit wide RANGEREDFRM variable was configured in the DVO per frame and the frame was scaled whenever RANGEREDFRM == 1. 0 = DISABLE 1 = ENABLE
22:20	0x0	HFTYPE: StretchBLT Horizontal Filter Mode (HFTYPE[2:0]) The six-tap horizontal interpolation filter can be operated in various operation modes. For the image expansion, it should be programmed as a pure 6-tap interpolator. For the image contraction, it can work as partly lowpass filter and partly interpolater with varying degree depending the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of horizontal filtering. 000: Pure interpolation filter 001, 010, 011, 100, 101, 110: mix of interpolation and low pass filters 111: DISABLE. 0 = INTERP 1 = LPF1 2 = LPF2 3 = LPF3 4 = LPF4 5 = LPF5 6 = LPF6 7 = DISABLE
19	0x0	DISCSC: enabled color space converter by default 0 = ENABLE 1 = DISABLE
18	0x0	VFEN: StretchBLT Vertical Filter Enable (VFEN) Vertical filter shall be disabled if source images come from host CPU via CPU Read-FIFO. (SISEL) Vertical filter may be disabled to save some memory bandwidth but this will likely result in degradation of image quality. This option may also be used in the case where source image is progressive scanning and there is no vertical scaling. 0 Vertical filter is disabled.1 Vertical filter is enabled. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
17:16	0x0	VFTYPE: StretchBLT Vertical Filter Mode (VFTYPE[1:0]). This bit is effective when the Vertical Filter is enabled. (bit 18) The two-tap vertical interpolation filter can be operated in various modes. For the image expansion, it should be programmed as a pure 2-tap interpolator. For the image contraction, it can work as partly averager and partly interpolator with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of vertical filtering. 00: Pure interpolation filter. 01: 25% averager, 75% interpolator. 10: 50% averager, 50% interpolator. 11: 100% averager. 0 = INTERP 1 = AVG25_INTERP75 2 = AVG50_INTERP50 3 = AVG
9:8	X	UVST: 00= 1/2 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 2 bytes. 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 4 bytes. 11= use uvstride. Note: Use care when using a tiled surface, since 1/2 or 1/4 of luma stride may yield an invalid pitch (stride). 0 = UVS2X 1 = UVS1X 2 = UVS4X 3 = UVS_G2UVSTRIDE
7	0x0	ENAHF: enable horizontal alpha filtering if disabled, use the alpha value of third tap for output pixel U,V line stride in 4:2:0 Format. 0 = DISABLE 1 = ENABLE
6	0x0	ENAVF:enable alpha vertical filter 0 = DISABLE 1 = ENABLE
5	0x0	IMODE: Source (input) Data Mode (IMODE). This parameter defines the data mode of source input. 0: Multiplexed Mode (data format is specified by SIFMT[2:0]). One block of source data in the Image Buffer memory contains multiplexed component signals. 1: Planar mode, 4:2:0 YUV (SIFMT[2] specifies data format of U/V components). Three blocks of source data in the Image Buffer memory contain separate Y, U, and V component signals. For a circular buffer input to gr2d, the input format cannot be planar. It must be multiplex, i.e., no YUV420 planar circular buffer input to gr2d. 0 = MULTIPLEX 1 = PLANAR
4	0x0	YUV422PLANAR: Source data is yuv422 planar valid when imode==PLANAR
3	0x0	YUV422ROTATION: YUV422 planar rotated.

14.7.25 G2SB_CTX5_G2CONTROLSECOND_0

Offset: 0x201e │ Byte Offset: 0x8078 │ Read/Write: R/W │ Reset: 0x00000000 (0bxx000000x00xxxx000xxxx000000000x)

Bit	Reset	Description
29	0x0	FR_READWAIT: Fast Rotate wait for reads. Enabling this bit forces FR to wait for the reads to be stored in the data return FIFO before sending writes out to the same block. Always set to ENABLE. This field is debug only. 0 = DISABLE 1 = ENABLE
28:26	0x0	FR_TYPE: Fast Rotate type 000 = FLIP_X 001 = FLIP_Y 010 = TRANS_LR (mirrors about diagonal. Diagonal runs from upper left to lower right) 011 = TRANS_RL (mirrors about diagonal. Diagonal runs from upper right to lower left) 100 = ROT_90 (counterclockwise by 90 degrees) 101 = ROT_180 110 = ROT_270 (clockwise by 90 degrees) 111 = IDENTITY 0 = FLIP_X 1 = FLIP_Y 2 = TRANS_LR 3 = TRANS_RL 4 = ROT_90 5 = ROT_180 6 = ROT_270 7 = IDENTITY
25:24	0x0	FR_MODE: Fast Rotate mode sel G2 Fast Rotate Transforms a surface via FR_TYPE transformation. Works in either 2-buffer (copy) or 1-buffer mode (in place) mode. The engine breaks down a larger surface into a grid of smaller FR_BLOCKs. Works on the granularity of an FR_BLOCK. An FR_BLOCK is: ▪ 16x16 pixel block (DSTCD = bpp8) ▪ 8x8 pixel block (DSTCD = bpp16) ▪ 4x4 pixel block (DSTCD = bpp32) Maximum surface size is 4096x4096 Key information - source and destination base address must be 128-bit word aligned - engine works on FR_BLOCK granularity: transformed surface width in multiples of 16 bytes** transformed surface height in multiples of 16/8/4 lines for bpp8/bpp16/bpp32 FR_BLOCK if surface dimension is not a multiple, software can program FR engine to transform larger surface (round up to next FR_BLOCK in width and height-->transform-->calculate relative memory pointer address) - during a rotational transformation (TRANS_LR, TRANS_RL, ROT_90, ROT_270): the stride of the output surface != the stride of the input surface when working on non-square input input: output: *^---------- *$---------- $ 0 1 2 3 4 5 6 7 ^ 24 16 8 0 ^ ^ ^ ^ | 8 9 10 11 12 13 14 15 | 25 17 9 1 ^ ^ ^ ^ |16 17 18 19 20 21 22 23 | 26 18 10 2 ^ ^ ^ ^ |24 25 26 27 28 29 30 31 | 27 19 11 3 ^ ^ ^ ^ | - - - - * * * * | 28 20 12 4 * * * * | - - - - * * * * | 29 21 13 5 * * * * | - - - - * * * * | 30 22 14 6 * * * * | - - - - * * * * | 31 23 15 7 * * * * - Tiling alignment restrictions subsume FR restrictions because the FR surface base address is the same as the start address. **Register Programming:** FR_MODE - inplace or copy FR_TYPE - type of transformation DSTCD - bpp8, bpp16, bpp32 SRCBA - source base address SRCWIDTH - (width in pixels-1) SRCHEIGHT - (height in lines-1) SRCS - source stride DSTBA - destination base address DSTS - destination stride FR_READWAIT - always set to enable FR inefficency in the following setup: 1. FR_MODE==SQUARE 2. if(FR_TYPE==YFLIP && SRCHEIGHT==m*n && n==odd number) where

Bit	Reset	Description
		m=8bpp?16:16bpp?8:4 Or if(FR_TYPE==XFLIP && SRCWIDTH==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 3. Then, the middle 8 lines/pixels, e.g., YFLIP/16bpp/n==3, line 8 to line 15 will be processed twice. 00 = disable fast rotate - this turns off the second level clock to fr_rotate engine. Please remember to do so! 01 = src/dst copy mode - two separate buffers 10 = square in place - one buffer 11 = blank 0 = DISABLE 1 = SRC_DST_COPY 2 = SQUARE 3 = BLANK
22:21	0x0	CLIPC: Clipping rectangle control, if clip enable, bit 57, XYTDW, should be cleared. 0x=clipping disabled, 10=draw only inside clipping rectangle, 11=draw only outside clipping rectangle
9	0x0	ALPSRCORDST: 32-bit blending mode, output alpha selection 0:source alpha, 1:destination alpha 0 = DISABLE 1 = ENABLE
8:4	0x0	ALPTYPE: Alpha blending method FIX: ALPHA blending: Fixed alpha, ALPHA is the value, B5G6R5 * B5G6R5 VCAA: B5G6R5->B5G6R5 PL1BPP: ALPHA blending: Alpha 1 bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero. PL2BPP: ALPHA blending: Alpha 2 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero. PL4BPP: ALPHA blending: Alpha 4 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero. PL8BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1 bit from source B5G5R5A1, alpha(MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4 bits from source A4B4G4R4, alpha(LSB). Dest: B5G6R5. PLS4BPP: ALPHA blending: Alpha 4 bits from source B4G4R4A4, alpha(MSB). Dest: B5G6R5. VCAA: R8G8B8A8->B5G6R5 without reading VCAA plane (surface blit with bpp down convert - implemented in hardware by the VCAA engine; not really a VCAA resolve) PLS8BPP: ALPHA blending: Alpha 8 bits from source/destination (decided by ALPSRCORDST), R8G8B8A8, alpha(MSB). DST: R8G8B8A8 VCAA: R8G8B8A8->R8G8B8A8, alpha has same blending method as RGB PLS8BX: ALPHA blending: Alpha 8 bits from source B8G8R8A8, alpha(MSB). Dest: B5G6R5. (**Restrictions) VCAA: R8G8B8A8->B5G6R5 PLS1BPPAL: ALPHA blending: Alpha 1 bit from source A1B5G5R5, alpha(LSB). Dest: B5G6R5 VCAA: A1B5G5R5->A1B5G5R5 **Restriction PLS8BX alpha blending has the following restrictions: 1. Source/destination addresses have to be in a 128-bit boundary. 2. Destination width has to be a multiple of 4 pixels. 3. Source/Destination strides have to be a multiple of 128 bits. 0 = FIX 1 = PL1BPP 2 = PL2BPP 3 = PL4BPP 4 = PL8BPP 5 = PL44BPP 6 = PLS1BPP 7 = PLS4BPPAL 8 = PLS4BPP

Bit	Reset	Description
		9 = PLS8BPP 10 = PLS8BX 11 = PLS1BPPAL
3	0x0	BEWSWAP: Host port word swap 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 0 = DISABLE 1 = ENABLE

14.7.26 G2SB_CTX5_G2CONTROLMAIN_0

G2 Blit Formats:

(1) G2 Blit size is based on height in lines, width in pixels, stride in bytes, and pixel size (color depth). Unless alpha blending is enabled, G2 does not care about pixel components.

Source color depths supported by G2 Blit:

- color depth same as destination: G2CONTROLMAIN.SRCCD = 1
- monochrome: G2CONTROLMAIN.SRCCD = 0

(source monochrome color determined by G2SRCBGC, G2SRCFGC)

Pattern color depths supported by G2 Blit:

- color depth same as destination: G2PATOS.PATCD = 1
- monochrome: G2PATOS.PATCD = 0

(pattern monochrome color determined by G2PATBGC, G2PATFGC)

Destination color depths supported by G2 Blit:

- 1 byte per pixel: G2CONTROLMAIN.DSTCD = BPP8
- 2 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP16
- 4 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP32

(2) Alpha blending operates on 3 or 4 component pixels of a specified color depth. One of the components may be alpha (A), depending on the alpha blend format, which is only allowed in certain component positions defined below. The only format rule of the remaining components is that they must be in the same source and destination position because the alpha blend engine does not support component swapping. For example, valid 32BPP formats are: RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888, GRBA_8888->GRBA_8888, etc.

Source formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16:
  - ALPTYPE_PLS1BPP: xxxA_5551 (for example, BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
  - ALPTYPE_PLS4BPP: xxxA_4444 (for example, BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
  - (other) : xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32:
  - (all) : xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

Destination formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16: xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

src cd	src format	dst cd	dst format	examples
BPP8	no support	BPP8	no support	
BPP16	xyzA_5551, xyzA_4444, xyz_565	BPP16	xyz_565	BGRA_5551->BGR_565, RGB_565->RGB565
BPP32	xyzA_8888	BPP32	xyzA_8888	RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888
BPP32	xyzA_8888	BPP16	xyz_565	RGBA_8888->RGB_565, BGRA_8888->BGR_565

VCAA Resolve Engine

(1) Format support

srcColor format	srcCoverage format	legal dest format(s)
A1B5G5R5	VCAA	A1B5G5R5
R8G8B8A8	VCAA	R8G8B8A8, B5G6R5
B5G6R5	VCAA	B5G6R5

To program resolve type:

- ALPTYPE == FIX: B5G6R5 -> B5G6R5
- ALPTYPE == PLS1BPPAL: A1B5G5R5 -> A1B5G5R5
- ALPTYPE == PLS8BPP: R8G8B8A8 -> R8G8B8A8
- ALPTYPE == PLS8BX: R8G8B8A8 -> B5G6R5

(2) Color surface programming

- color surface base address: SRCBA
- subsurface start x: SRCX (pixel index)
- subsurface start y: SRCY (line index)
- color surface stride: SRCS (bytes)

The engine supports subsurface resolve. One can imagine a full color buffer of 1024x768, but

```
[pseudo_code]
      color_surface_start_address_fetch = pcurctx->regs.rG2SRCBA.uSRCBA() +
       (pcurctx->regs.rG2SRCPS.uSRCX() * VCAAState.color_surface_depth) +
       (pcurctx->regs.rG2SRCPS.uSRCY() * VCAAState.color_surface_stride);
```

(3) VCAA surface programming

- VCAA surface base address: PATBA
- VCAA surface stride: PATST (bytes)

```
[pseudo_code]
      vcaa_surface_start_address_fetch = pcurctx->regs.rG2PATBA.uPATBA();
```

Software *MUST* directly program this register to the proper subsurface location in the VCAA surface which corresponds to the SRCX and SRCY programming. The calculation is PATBA = vcaa_base_address + SRCX * (1) + SRCY * PATST;

(4) Resolve surface programming

- resolve window width: DSTWIDTH (pixels)
- resolve window height: DSTHEIGHT (lines)
- resolve output stride: DSTS (bytes)

These values program the width and height of the resolved surface. In the base case:

- DSTWIDTH = color surface width
- DSTHEIGHT = color sufrace height
- DSTS = resolve surface format bpp * DSTWIDTH

The VCAA engine technically supports DSTX and DSTY (like SRCX and SRCY).

```
[pseudo_code]
   resolve_surface_start_address_put = pcurctx->regs.rG2DSTBA.uDSTBA() +
     (pcurctx->regs.rG2DSTPS.uDSTX() * resolve_surface_color_depth) +
     (pcurctx->regs.rG2DSTPS.uDSTY() * resolve_surface_color_stride);
```

(5) Resolving a pixel

The coverage surface is an 8 bits per pixel surface (C4X4). The surface is initialized to all 1s by 3d.

For a given pixel, we need to look up 4 bits of coverage data. If the data is 0xf, the pixel does not need to be resolved. Otherwise, the pixel needs to be reblended with certain weightings of its neighbors (i.e., we are on the edge of some geometry).

If any coverage bits are 0, the color needs to be resolved. To do so, the following equation is calculated:

```
color_new = (
              20*color_old +
              27*(
                  ((cover_down  == 0) ? color_down  : color_old) +
                  ((cover_right == 0) ? color_right : color_old) +
                  ((cover_left  == 0) ? color_left  : color_old) +
                  ((cover_up    == 0) ? color_up    : color_old)
                  )
            ) / 128;
```

If coverage bit == 0, then use the neighbor color value in the resolve.

If coverage bit == 1, then use the center color value in the resolve.

The resolve is done on a per channel (r/g/b/a) basis:

format convert input channel to 8 bits

perform resolve equation above

format convert to output format

For the resolution of the alpha channel:

- In B5G6R5: No alpha information. Do nothing.
- In A1B5G5R5: No resolve. Keep the source color alpha bit.
- In R8G8B8A8: Normal 8-bit channel resolve

(6) Resolving a pixel on the edge

When on an edge, the VCAA engine treats the coverage bit as a 1. Use the center color value. This holds true even on the edge of the subsurface resolve window when located in the middle of the larger surface.

(7) Maximum dimensions

The maximum resolve subsurface is 4096 pixels wide.

(8) Surface restrictions

The VCAA engine has the following restrictions with respect to input surfaces:

- input base address is 128-bit memory word aligned
- input stride is a multiple of 128 bits (16 bytes)

The VCAA engine has the following restrictions with respect to output surfaces

- output base address is 128-bit memory word aligned
- output stride is a multiple of 128 bits (16 bytes)

(9) Coverage surface layout

The surface is C4X4.

Surface layout (C4X4)

```
(lsb2msb)  C0_C1_C2_C3_X_X_X_X     (X is a don't care.  Qrast inits to 1)
(msb2lsb)  X_X_X_X_C3_C2_C1_C0


    +          o          o
         C0
            C1   Cov(0,1)
       C2
         C3
    o          o          o

      Cov(1,0)   Cov(1,1)


    o          o          o


    +/o = pixel centers
```

In the example above,

```
       Cov(0,0) = X_X_X_X_C3_C2_C1_C0
       C3       = V_C3(Cov(0,0));
       ..         ..
       C0       = V_C0(Cov(0,0));
    #define V_C3(bits) (bits & 0x8)
    #define V_C2(bits) (bits & 0x4)
    #define V_C1(bits) (bits & 0x2)
    #define V_C0(bits) (bits & 0x1)
```

Resolving a pixel

In an ideal world, the coverage would be simply laid out such that when resolving Pixel(x,y), we would need only to fetch Cov(x,y) to get all 4 bits of information.

In actuality, the coverage surface was rearranged to help qrast in performance. To resolve Pixel(x,y) gr2d has to fetch 4 C4X4 bytes [Cov(x,y), Cov(x-1,y), Cov(x,y-1), Cov(x-1,y-1)]. From each C4X4 byte, it extracts one bit which is used in the resolve equations shown in (5).

Need the logical vcaa bits (DRLU) for blend which is spread across four C4X4 coverage bytes

```
          U

    L

        o

            R

      D
```

where

o = Pixel(x,y)

D = V_C1(Cov(x-1,y))

R = V_C0(Cov(x,y))

L = V_C3(Cov(x-1,y-1))

U = V_C2(Cov(x,y-1))

Offset: 0x201f | Byte Offset: 0x807c | Read/Write: R/W | Reset: 0x00000000 (0bx000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: pattern Data Select: SRCSEL and PATSEL cannot be both enabled. 0 = SCREEN 1 = MEMORY
28	0x0	DSTDIR:destination direct addressing
27	0x0	SRCDIR:source direct addressing
26:25	0x0	GCSW: Display Switching Window Control (GCSW[1:0]). This parameter controls multi-buffering for Display. x0: At end of current command, do not send signal to Display to switch buffer. 01: two buffers, dstba and dstba_b are used 11: three buffers, dstba, dstba_b, dstba_c are used
24	X	SRCBAS: Source base address select: 0=srcba 1=dstba This is not used for Line Draw and if source data comes from host memory.
23:22	0x0	SRCT: Source transparency enable: 0x=source transparency disabled, 10=mono source background transparency or color source transparency, 11=mono source foreground transparency or inverse color source transparency. NOTE: When source transparency is enabled and SRCCD==0(mono) SRCBGC!=SRCFGC should be satisfied.
21	X	HLMONO: Start from Msb or lsb in byte when mono expansion. If HLMONO is 1, bit 0 (the lsb) is the first bit. If HLMONO is 0, bit 7 (the msb) is the first bit
20	X	SRCCD:

Bit	Reset	Description
		0: Source mono 1: Source has same color depth as destination SRCCD==0 xdir/ydir has to be 0
19:18	0x0	DSTT: Destination read transparency enable: 0x=destination read transparency disabled 10=color destination read transparency 11=inverse color destination read transparency. 20 rw INIC Initiate Command (1=initiate command immediately, 0=wait for launch write)
17:16	X	DSTCD: Destination color depth: 00=8-bpp, 01=16-bpp, 10=32-bpp. 11=reserved 0 = BPP8 1 = BPP16 2 = BPP32 3 = RESERVED3
15	0x0	SRCSEL: Source Data Select: 0 = SCREEN 1 = MEMORY
14	0x0	YFLIP: flip y direction to make image upside down or the other way. If YFLIP==1, ROP cannot include destination. 0 = DISABLE 1 = ENABLE
13	0x0	PATPACK: Pattern Data is in Pack Mode. PATLNGAP in G2PATPACK is the line gap for pattern packed mode If (PATPACK && ~PATSEl), pattern data is packed and from screen, PATMONOW/PATMONOH should be programmed properly to fetch pattern data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
12	0x0	SRCPACK: Source Data is in Pack Mode. SRCLNGAP in G2SRCPACK is the line gap for source packed mode. If (SRCPACK && ~SRCSEL), source data is packed and from screen, SRCMONOW/SRCMONOH should be programmed properly to fetch data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
11	0x0	XYTDW: xy transpose, Line stride DSTS has to be 16-byte aligned if enabled. If XYTDW==1, ROP cannot include destination.
10	0x0	YDIR: 0=incrementing, 1=decrementing. ydir should not be set when source surface has different color depth as destination surface. SRCCD==0 (mono src) OR PATCD==0 (mono pat), ydir has to be 0
9	0x0	XDIR: 0=incrementing, 1=decrementing. xdir should not be set when source surface has different color depth as destination surface.For example, ALPTYPE=PLS8BX (32bpp blending with 16bpp), xdir has to be 0 SRCCD==0 (mono src) OR PATCD==0 (mono pat), xdir has to be 0.
8	0x0	PATFL: When mono pattern is set, we use mono tile pattern fill. current pattern client can support a 16x16 bit tile which can be used to generate a larger surface (i.e. tile this surface in the x and y direction) The tile is stored in memory. Currently, PATXO and PATYO indicate where to start in the 16x16 tile when expanding the monochrome data. See patxo and patyo comments below for programming. When xdir==1 Patxo = pattern width - (patxo+destination width)&0xF -- For xoffsets y offsets PATYO remains the same when xdir/ydir change. How the tile replication pattern looks based on xdir/ydir (without programming patxo above): xdir=0 ydir=0 xdir=1 ydir=0 +---+--+ +--+---+ |^|^|^|| ||^|^|^|

Bit	Reset	Description
		|~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ // xdir=0 ydir=1 xdir=1 ydir=1 Mono tile is +---+ |^|^| |~~~| +---+ 0 = DISABLE 1 = ENABLE
7	0x0	PATSLD: BitBlt Solid Pattern Fill: 1=enable. BGC will be used as the color value. 0 = DISABLE 1 = ENABLE
6	0x0	SRCSLD: BitBlt Solid Source Color Fill: 1=enable. FGC will be used as the color value. 0 = DISABLE 1 = ENABLE
5	0x0	ALPEN: BitBlt Alpha Blending, 1=enable. 0=disable,when both Faden and alpen are 1, output=Source*alpha_v + fadoff, alpha_v is decided by alptype
4	0x0	FADEN: BitBlt Source Copy Fade enable, 1=enable (share with mltln), only support 16bpp mode 0 = DISABLE 1 = ENABLE
3	0x0	TEST0BIT:Command finish timing bit 0: 2D command finishes when last data has been pushed to memory write client. 1: 2D command waits memory write client to be idle to finish. 0 = DISABLE 1 = ENABLE
2	0x0	TURBOFILL:fast fill rectangle in 128 bits/clock. Some limitations with this mode:srcsld==1 rop==0xcc, no clipping, no transparency, xdir==0, ydir==0, flip==0, xytdw==0. Results are undefined if the above limitations are not satisfied.
1:0	0x0	CMDT: Command Type: 00=BitBlt 01=Line Draw 10=VCAA 11=reserved When the raise command is executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

14.7.27 G2SB_CTX5_G2ROPFADE_0

Offset: 0x2020 | Byte Offset: 0x8080 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:24	X	FADOFF:
23:16	X	FADCOE:
7:0	0x0	ROP: If YFLIP==1 or XYTDW==1, ROP cannot include destination.Since destination may have been corrupted before reading out.

14.7.28 G2SB_CTX5_G2ALPHABLEND_0

Offset: 0x2021 | Byte Offset: 0x8084 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description

Bit	Reset	Description
31:24	X	ALPHA1V
23:16	X	ALPHA0V
8	0x0	ALPHAINV
7:0	X	ALPHA

14.7.29 G2SB_CTX5_G2CLIPLEFTTOP_0

ClipRect Methods. Top-left bounds are inclusive

Offset: 0x2022 | Byte Offset: 0x8088 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPT
14:0	X	CLIPL

14.7.30 G2SB_CTX5_G2CLIPRIGHTBOT_0

Bottom-right bounds are exclusive

Offset: 0x2023 | Byte Offset: 0x808c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPB
14:0	X	CLIPR

14.7.31 G2SB_CTX5_G2PATPACK_0

Pattern packed mode

Pattern methods: G2PATPACK should be used to specify the line gap. Use G2PATPACK_SIZE to program height and width PACK is generally only useful with narrow monochrome surfaces

> **Note:** PACK is not officially supported when the surface is tiled.

Offset: 0x2024 | Byte Offset: 0x8090 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

14.7.32 G2SB_CTX5_G2PATPACK_SIZE_0

Pattern packed mode

Packed mode, pattern data line gap. Byte

G2PATPACK_SIZE -- extension of G2PATPACK, this register only holds the size of the packed data

> **Note:** PACK is not officially supported when the surface is tiled.

Offset: 0x2025 | Byte Offset: 0x8094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	PATMONOH: Pattern mono data height in packed mode, byte
15:0	X	PATMONOW: Mono data width in packed mode, byte

14.7.33 G2SB_CTX5_G2PATBA_0

Offset: 0x2026 | Byte Offset: 0x8098 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBA: If (PATFL==1){ It has to be 16-byte aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

14.7.34 G2SB_CTX5_G2PATOS_

Offset: 0x2027 | Byte Offset: 0x809c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00xxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	PATYO: y offset for mono tile pattern fill. see PATFL
27:24	X	PATXO: x offset for mono tile pattern fill. see PATFL
22:21	0x0	PATT: Mono pattern transparency enable: 0x=pattern transparency disabled, 10=mono pattern background transparency or color pattern transparency, 11=mono pattern foreground transparency or inverse color pattern transparency. NOTE: When pattern transparency is enabled and PATCD==0(mono) PATBGC!=PATFGC should be satisfied.
16	X	PATCD:0 mono1 same as dstcd PATCD==0 xdir/ydir has to be 0
15:0	X	PATST:stride

14.7.35 G2SB_CTX5_G2PATBGC_0

Offset: 0x2028 | Byte Offset: 0x80a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

14.7.36 G2SB_CTX5_G2PATFGC_0

Offset: 0x2029 | Byte Offset: 0x80a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

14.7.37 G2SB_CTX5_G2PATKEY_0

Offset: 0x202a | Byte Offset: 0x80a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

14.7.38 G2SB_CTX5_G2DSTBA_0

Offset: 0x202b | Byte Offset: 0x80ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA: Destination base address (byte address)

14.7.39 G2SB_CTX5_G2DSTBA_B_0

Offset: 0x202c | Byte Offset: 0x80b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_B: Destination base address (byte address) only usable in hardware trigger mode by enable gcsw

14.7.40 G2SB_CTX5_G2DSTBA_C_0

Offset: 0x202d | Byte Offset: 0x80b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_C: Destination base address (byte address)

14.7.41 G2SB_CTX5_G2DSTST_0

Offset: 0x202e | Byte Offset: 0x80b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

14.7.42 G2SB_CTX5_G2SRCPACK_0

Source Data Packed Mode

Surface methods. G2SRCPACK should only be used to specify the line gap use G2SRCPACK_SIZE to program height and width

Offset: 0x202f | Byte Offset: 0x80bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

14.7.43 G2SB_CTX5_G2SRCPACK_SIZE_0

Source Data Packed Mode

In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, the required width/stride is 16 byte aligned. Packed mode - source mono data line gap

G2SRCPACKS_SIZE -- extension of G2SRCPACK, this register only holds the size of the packed data

Offset: 0x2030 | Byte Offset: 0x80c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description

Bit	Reset	Description
31:16	X	SRCMONOH: Packed mode - source mono data height in bytes
15:0	X	SRCMONOW: Packed mode - source mono data width in bytes

14.7.44 G2SB_CTX5_G2SRCBA_0

Source Base Address

Offset: 0x2031 | Byte Offset: 0x80c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA: Source base address (byte address)

14.7.45 G2SB_CTX5_G2SRCBA_B_0

SB only

Offset: 0x2032 | Byte Offset: 0x80c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA_B: This parameter specifies the start address of source image stored in the image buffer memory. In 4:2:0 format mode, this image block accommodates for Y-image.This address specifies byte position; however, bits [2:0] are restricted with respect to the data formats to fit multiple pixels in one memory word (8 bytes). For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format. Since one Y pixel takes 8 bits, all 8 byte positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there are no restrictions on this value.)

14.7.46 G2SB_CTX5_G2SRCST_0

Offset: 0x2033 | Byte Offset: 0x80cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS: Source Stride coordinate (bytes) with respect to SRCBA. In order to fit multiple pixels in one memory word (8 bytes), bits [2:0] are restricted with respect to the data formats. For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format.

14.7.47 G2SB_CTX5_G2SRCBGC_0

srcColors

Offset: 0x2034 | Byte Offset: 0x80d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

14.7.48 G2SB_CTX5_G2SRCFGC_0

Offset: 0x2035 | Byte Offset: 0x80d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

14.7.49 G2SB_CTX5_G2SRCKEY_0

Offset: 0x2036 | Byte Offset: 0x80d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

14.7.50 G2SB_CTX5_G2SRCSIZE_0

Offset: 0x2037 | Byte Offset: 0x80dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	SRCHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	SRCWIDTH

14.7.51 G2SB_CTX5_G2DSTSIZE_0

Offset: 0x2038 | Byte Offset: 0x80e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	DSTHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	DSTWIDTH: In 2D mode, the largest number can be programmed is 0x7fef

14.7.52 G2SB_CTX5_G2SRCPS_0

ImageBlit Methods

Offset: 0x2039 | Byte Offset: 0x80e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCY
15:0	X	SRCX:SRCX[2:0] are ignored in SRCCD==0 (mono expansion). The first bit of the first byte (bit 7 if HLMONO==0, or bit 0 if HLMONO==1) always expand to DSTX,DSTY

14.7.53 G2SB_CTX5_G2DSTPS_0

Offset: 0x203a | Byte Offset: 0x80e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	DSTY
15:0	X	DSTX: NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero

14.7.54 G2SB_CTX5_G2CBDES_0

Circular Buffer

Offset: 0x203b | Byte Offset: 0x80ec | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TOPCLIP:top clipping at the first buffer, (buffer start) refer to TOP_CBLINE in G2CBDES2 0 = DISABLE 1 = ENABLE
30:16	X	CBLINE:vertical line number in one buffer
7:0	X	CBCOUNT: This specifies the number of buffers in circular buffer feature

14.7.55 G2SB_CTX5_G2CBSTRIDE_0

Offset: 0x203c | Byte Offset: 0x80f0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	X	CBUVSTRIDE: Chroma Buffer Stride default is half of luma 00= 1/2 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 2 bytes. 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 4 bytes. 1x= Reserved 0 = CBS2X 1 = CBS1X 2 = CBS4X
23:0	X	CBSTRIDE: Video Buffer Luma(or RGB) Buffer Stride. This is luma buffer stride (in bytes)

14.7.56 G2SB_CTX5_G2LINESETTING_0

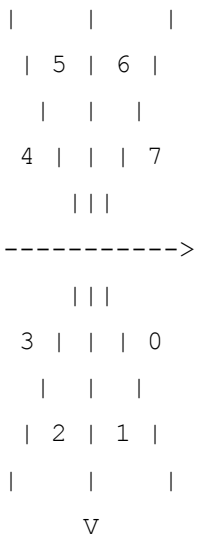
Line Methods

Offset: 0x203d | Byte Offset: 0x80f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	OCTANTS: 000: octant 0 001: octant 1 010: octant 2 011: octant 3 100: octant 4 101: octant 5 110: octant 6 111: octant 7
28	X	LINEUSEOCTANT:use OCTANTS in G2LINEDELTAN register instead of MAJOR LINEXDIR LINEYDIR
27	X	DROPLASTP:draw last pixel or not
26	X	LINEYDIR
25	X	LINEXDIR

Bit	Reset	Description
24	X	MAJOR: 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

14.7.57 G2SB_CTX5_G2LINEDELTAN_0



Offset: 0x203e | Byte Offset: 0x80f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

14.7.58 G2SB_CTX5_G2LINEDELTAM_0

Offset: 0x203f | Byte Offset: 0x80fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

14.7.59 G2SB_CTX5_G2LINEPOS_0

Offset: 0x2040 | Byte Offset: 0x8100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LINEYPOS
15:0	X	LINEXPOS

14.7.60 G2SB_CTX5_G2LINELEN_0

Offset: 0x2041 | Byte Offset: 0x8104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

14.7.61 G2SB_CTX5_G2CSCFOURTH_0

Offset: 0x2042 | Byte Offset: 0x8108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:16	X	G2V: Multiplier for G for V generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.368 (decimal) or 0x12F. For any other combination, this parameter is ignored.
8:0	X	G2U: Multiplier for G for U generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.291 (decimal) or 0x125. For any other combination, this parameter is ignored.

14.7.62 G2SB_CTX5_G2SRCST_B_0

Offset: 0x2043 | Byte Offset: 0x810c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

14.7.63 G2SB_CTX5_G2UVSTRIDE_0

Offset: 0x2044 | Byte Offset: 0x8110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

14.7.64 G2SB_CTX5_G2CBDES2_0

Circular Buffer Controller 2

Offset: 0x2045 | Byte Offset: 0x8114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

14.7.65 G2SB_CTX5_G2TILEMODE_0

Offset: 0x2046 | Byte Offset: 0x8118 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx0xxx0xxx0xxx0xxx0xxx0)

Bit	Reset	Description
20	0x0	DST_WR_TILE_MODE:destination surface 0 = LINEAR 1 = TILED
16	0x0	DST_RD_TILE_MODE:Same as destination write unless DST_RD_WR_SEP (not supported) 0 = LINEAR

Bit	Reset	Description
		1 = TILED
12	0x0	PAT_UV_TILE_MODE:UNUSED 0 = LINEAR 1 = TILED
8	0x0	PAT_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED
4	0x0	SRC_UV_TILE_MODE:UV surface, ignored in RGB mode 0 = LINEAR 1 = TILED
0	0x0	SRC_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED

14.7.66 G2SB_CTX5_G2PATBASE_0

Offset: 0x2047 │ Byte Offset: 0x811c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PAT_BASE: pattern base address in tile mode, PATBA is the linear address where pixel start

14.7.67 G2SB_CTX5_G2SRCBA_SB_SURFBASE_0

SB_SURFBASE Registers

These registers need only be programmed when using SB. They point to the base of the various source and destination surfaces. Technically they are only needed when tiling is enabled, but there is no harm in always programming them.

Their counterpart registers (_SB_SURFBASE with SB_SURFBASE stripped off) indicate the location of the first pixel to be sourced or written within the surface.

This register exists to mimic the X, Y (SRCX, SRCY) functionality found in the BitBlt engine.

For example, to get at pixel X, Y:

SRCBA = SRCBA_SB_SURFBASE + Y*stride + X*Bpp

Offset: 0x2048 │ Byte Offset: 0x8120 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRC_ADDR: surface address corresponding to G2SRCBA: -base of interleaved sources (RGB, YUV) -base of Y plane. Only used by the StretchBlit Engine

14.7.68 G2SB_CTX5_G2DSTBA_SB_SURFBASE_0

Offset: 0x2049 │ Byte Offset: 0x8124 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_ADDR: surface address corresponding to G2DSTBA. Only used by the StretchBlit Engine

14.7.69 G2SB_CTX5_G2DSTBA_B_SB_SURFBASE_0

Offset: 0x204a | Byte Offset: 0x8128 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_B_ADDR: surface address corresponding to G2DSTBA_B Only used by the StretchBlit Engine, and G2CONTROLSB.DBSEL() is enabled

14.7.70 G2SB_CTX5_G2VBA_A_SB_SURFBASE_0

Offset: 0x204b | Byte Offset: 0x812c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	V_ADDR: surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of V plane. Only used by the StretchBlit Engine

14.7.71 G2SB_CTX5_G2UBA_A_SB_SURFBASE_0

Offset: 0x204c | Byte Offset: 0x8130 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	U_ADDR: surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of U plane. Only used by the StretchBlit Engine

14.8 G2SB CTX 6 Registers

14.8.1 G2SB_CTX6_INCR_SYNCPT_0

Memory Controller Tiling Definitions

Offset: 0x6000 | Byte Offset: 0x18000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	INDX: syncpt index value

14.8.2 G2SB_CTX6_INCR_SYNCPT_CNTRL_0

Offset: 0x6001 | Byte Offset: 0x18004 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

14.8.3 G2SB_CTX6_INCR_SYNCPT_ERROR_0

Offset: 0x6002 | Byte Offset: 0x18008 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

14.8.4 G2SB_CTX6_G2CLASSCHANNEL_REGONLY_0

This register can be accessed by both register read/write and command FIFO in order not to break exsting code, keep suffix REGONLY.

2D syncpt behavior

- Each 2D context has its own syncpt client.
- OP_DONE increments when 2D's memory write client is idle, meaning all preceding operations are complete.
- RD_DONE behaves like OP_DONE
- REG_WR_SAFE behaves like IMMEDIATE meaning a syncpt increment returns immediately and does not indicate that a register write will NOT block. The implications are that 2D's host FIFO can fill and block access to 2D, which will indeterminately block a direct CPU access.

Offset: 0x6008 | Byte Offset: 0x18020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

14.8.5 G2SB_CTX6_G2TRIGGER_0

Control methods, since they are not programmed often, put them in the beginning.

Offset: 0x6009 | Byte Offset: 0x18024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

14.8.6 G2SB_CTX6_G2TRIGGER1_0

Offset: 0x600a | Byte Offset: 0x18028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

14.8.7 G2SB_CTX6_G2TRIGGER2_0

Offset: 0x600b | Byte Offset: 0x1802c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

14.8.8 G2SB_CTX6_G2CMDSEL_0

Offset: 0x600c | Byte Offset: 0x18030 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxx0xxxxxxxx00x0x0000000xxxx)

Bit	R/W	Reset	Description
28	RW	0x0	PRIORITY: indicates priority of the context, note VI-triggered contexts take priority over HIGH 0 = LOW 1 = HIGH
27:25	RW	X	LINKVAL
24	RW	0x0	LINKEN:Test purpose
23:16	RW	X	BUFFER_INDEX
15	RW	0x0	FRAME_END
14	RW	0x0	FRAME_START:host writes this bit to tell which buffer is ready
10	RW	0x0	CLIP_SOURCE_TOP_BOTTOM: Reserved for hardware test. Software should not use this bit. 0 = DISABLE 1 = ENABLE
9:8	RW	0x0	G2OUTPUT: 00: to memory 01: to EPP 0 = MEMORY 1 = EPP 2 = RESERVED2 3 = RESERVED3
7	RW	0x0	CBSBDISABLE
6	RW	0x0	HOSTTRIGGER:host trigger the command, host also need program circular buffer related data
5	RW	0x0	VITRIGGER:VI trigger enable, can be circular buffer or full frame
4	RW	0x0	CBENABLE:circular buffer feature enable 0 = DISABLE 1 = ENABLE
0	RO	X	SBOR2D: 0 = G2 1 = SB

14.8.9 G2SB_CTX6_G2RAISE_0

Offset: 0x600d | Byte Offset: 0x18034 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE:Push back to read FIFO when all commands in the channel are done.

14.8.10 G2SB_CTX6_G2HOSTSET_0

Offset: 0x600f | Byte Offset: 0x1803c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	HSTFPXL: specifies the first pixel location (in byte) in the first source data entry. In mono mode pixel should always start on a byte boundary.
3:0	0x0	HSTLNGAP: specifies distance (in byte) from last pixel of a line to the first pixel of next line. In mono mode, the value is the nearest byte number. For example, if gap is 3 bits, HSTLNGAP should be 1 (byte), if gap is 9 bits, HSTLNGAP should be 2 (byte).

14.8.11 G2SB_CTX6_G2HOSTFIFO_0

Offset: 0x6010 | Byte Offset: 0x18040 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

14.8.12 G2SB_CTX6_G2VDDA_0

Offset: 0x6011 | Byte Offset: 0x18044 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VDSTEP: Vertical Scaling DDA. Reset value: xxxx-xxxxh. Vertical DDA Step (Increment) Value (VDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for vertical scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_height-1-VDTINI) / (Actual_destination_height-1). Truncate the remaining bits to keep the 12-bit fraction. Since we have to meet (Actual_source_height-1)*1.0 >= (Actual_destination_height - 1)*VDSTEP + VDTINI. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction enensures the maximum limit of accumulated error less than half line-spacing for up to approximately 2000 target lines. For example, image expansion from 240 lines to 400 lines VDSTEP[17:0] = 19'b00_0000_1001_1001_1010 and image contraction from 240 lines to 150 lines VDSTEP[17:0] = 19'b00_0001_1001_1001_1010.

14.8.13 G2SB_CTX6_G2VDDAINI_0

Offset: 0x6012 | Byte Offset: 0x18048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	VDBINI:Vertical DDA Initial Value for Bottom-Field (VDBINI[7:0]) (see description above)
7:0	X	VDTINI: Vertical Scaling DDA Initial Values. Vertical DDA Initial Value for Top-Field (VDTINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for vertical scaling.

Bit	Reset	Description
		Given a pair of consecutive source image lines representing positions 0.0 and 1.0, a destination line-image will be created at any position in [0.0, 1.0), that is specified by the fraction part of the vertical DDA. The 8-bit initial fraction value specifies the position of the first output (destination) line-image located between the first and the second input (source) line-images, respectively, at position 0.0 and 1.0. This parameter provides a way to compensate relative displacement of top and bottom fields of a source image. Suppose, 2-to-1 interlaced original video field images #1 and #2 are in the image buffer, to be displayed in an overlay window. Assume that the #1 field image is the top-field. Using VDTINI = 8'hC0 for the top-field image and VDBINI = 8'h40 for the bottom-field image, for example, the two images from #1 and #2 fields get mapped to identically positioned destination images in the overlay window.

14.8.14 G2SB_CTX6_G2HDDA_0

Offset: 0x6013 │ Byte Offset: 0x1804c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HDSTEP: Horizontal Scaling DDA. Reset value: xxxx-xxxxh. Horizontal DDA Step (Increment) Value (HDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for horizontal scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_width-1-HDINI) / (Actual_destination_width-1). Truncate the remaining bits to keep the 12-bit fraction. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half pixel-spacing for up to approximately 2000 destination pixels. For example, image expansion from 720 pixels to 800 pixels HDSTEP[17:0] = 19'b000_0000_1110_0110_0111 and image contraction from 720 pixels to 250 pixels HDSTEP[17:0] = 19'b000_0010_1110_0001_0101.

14.8.15 G2SB_CTX6_G2HDDAINILS_0

Offset: 0x6014 │ Byte Offset: 0x18050 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	HDINI: Horizontal Scaling DDA Initial Value. Horizontal DDA Initial Value (HDINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for horizontal scaling.Given a group of six consecutive source pixels that the two pixels at the center representing positions 0.0 and 1.0, a destination pixel will be created at any position in [0.0, 1.0), that is specified by the fraction part of the horizontal DDA. The 8-bit initial fraction value specifies the position of the first output (destination) pixel located between the first and the second input (source) pixels, respectively at position 0.0 and 1.0.For horizontal scaling this value may normally be set to 0.

14.8.16 G2SB_CTX6_G2CSCFIRST_0

The next 3 registers contain 8 parameters used by the YCbCr (YUV) to RGB color space conversion logic. This logic can also be used for RGB gain/gamma correction in RGB to RGB modes, and for luma gain in YUV to YUV modes.

Two new parameters (G2U, G2V) have been added for RGB to YUV conversion. These are in address 0x38. The equations have the form:

if (yuv2rgb)

(r,g,b) = matrix*(y+sbreg_yos,u,v)

matrix: sbreg_cyx, sbreg_cur, sbreg_cvr,
sbreg_cyx, sbreg_cug, sbreg_cvg,
sbreg_cyx, sbreg_cub, sbreg_cvb,

else if (rgb2rgb)

(r,g,b) = matrix*(r,g+sbreg_yos,b)

matrix: sbreg_cvr, 0, sbreg_cur,
sbreg_cvg, sbreg_cyx, sbreg_cug,
sbreg_cvb, 0, sbreg_cub,

else if (yuv2yuv)

(y,u,v) = matrix*(y,u,v)

matrix: sbreg_cub, 0, sbreg_cvb,
sbreg_cug, sbreg_cyx, sbreg_cvg,
sbreg_cur, 0, sbreg_cvr,

else if (rgb2yuv)

(y,u,v) = matrix*(r,g,b) + (sbreg_yos,0,0)

matrix: sbreg_cvb, sbreg_cyx, sbreg_cub,
sbreg_cvg, sbreg_g2u, sbreg_cug,
sbreg_cvr, sbreg_g2v, sbreg_cur,

Offset: 0x6015 | Byte Offset: 0x18054 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	YOS: Y-Offset (YOFFSET[7:0]) for YUV generation. This parameter consists of 8-bit 2's complement in the range [-128,127]. For YUV->RGB, the recommended value is -16 (decimal) or 0xF0. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is +16 (decimal) or 0x10
21:12	X	CVR: multiplier for V/R for V or R generation.This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 1.5960 (decimal) or 0x0CCIf source data is in RGB format, this parameter may be used as gain adjustment for R component. For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038
9:0	X	CUB: multiplier for U/B for Y or B generation. Consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 2.0172 (decimal) or 0x102. If source data is in RGB format, this parameter may be used as gain adjustment for B component. This register changes precision when doing RGB to YUV conversion (SIFMT=1xxx, DIFMT=0xxx). CUB becomes s1.8 and the recommended value is +0.098 or 0x019

14.8.17 G2SB_CTX6_G2CSCSECOND_0

Offset: 0x6016 | Byte Offset: 0x18058 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	CYX: multiplier for Y/G (G gain). This positive-only parameter consists of 8-bit magnitude (1.7). For YUV->YUV, the recommended value is 1.1644 (decimal) or 0x95. For YUV->YUV or RGB->RGB, this may be used as gain adjustment for Y or G component. For RGB->YUV, the recommended value is +0.504 (decimal) or 0x041
21:12	X	CUR: multiplier for U/B for V or R generation.This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This parameter takes non-zero value if hue is rotated. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.071 (decimal) or 0x209

Bit	Reset	Description
8:0	X	CUG: multiplier for U/B for U or G generation. Consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB, the recommended value is -0.3918 (decimal) or 0x132. For RGB->RGB, this parameter should be set to 0. For YUV->YUV, this parameter should be set to 1 (0x080). For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038

14.8.18 G2SB_CTX6_G2CSCTHIRD_0

Offset: 0x6017 | Byte Offset: 0x1805c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	CVB: Multiplier for V/R for Y or B generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This coefficient takes a non-zero value if the hue is rotated. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is +0.257 (decimal) or 0x021
8:0	X	CVG: Multiplier for V/R for U or G generation.This parameter consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB, the recommended value is -0.8130 (decimal) or 0x168. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.148 (decimal) or 0x113

14.8.19 G2SB_CTX6_G2CMKEYL_0

When key generation is enabled, the value of this register is the lower color/chroma limit and G2CMKEYU is the upper color/chroma limit for component pixel data. Three component signal values (YCrCb/YUV or RGB) of every output pixel are compared to a set of three ranges that are specified by three pairs of lower and upper color/chroma key values.

If the Key Polarity is 0, the Key is set to 1 only when all three component values of an input pixel are respectively within the set of ranges (inclusive of the limit values), else the Key is set to 0.

If the Key Polarity is 1, the Key is set to 1 when any of the three component values of an input pixel are outside the set of ranges, else the Key is set to 0.

StretchBLT, Color/Chroma Key Lower Limit Register

Offset: 0x6018 | Byte Offset: 0x18060 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRL: R or Cr Color Chroma Key Lower Limit (CKRL[7:0]) Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGL: G or Cb Color Chroma Key Lower Limit (CKGL[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
7:0	X	CKBL: B or Y Color/Chroma Key Lower Limit (CKBL[7:0])

14.8.20 G2SB_CTX6_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0x6019 | Byte Offset: 0x18064 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRU: R or Cr Color Chroma Key Upper Limit (CKRU[7:0]) Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal

Bit	Reset	Description
		level to the maximum signal level.
15:8	X	CKGU: G or Cb Color/Chroma Key Upper Limit (CKGU[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity the minimum signal level to the maximum signal level.
7:0	X	CKBU: This is B or Y color/chroma key upper limit value.

14.8.21 G2SB_CTX6_G2UBA_A_0

StretchBLT, Source Image Data U base address A-buffer

Offset: 0x601a │ Byte Offset: 0x18068 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SU1SA: Start Address of Source U-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.8.22 G2SB_CTX6_G2VBA_A_0

Offset: 0x601b │ Byte Offset: 0x1806c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SV1SA: Start Address of Source V-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.8.23 G2SB_CTX6_G2SBFORMAT_0

Offset: 0x601c │ Byte Offset: 0x18070 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12:8	X	DIFMT: Destination Image Data Format This parameter defines the data format of destination output. There are two groups of data formats, RGB and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement. 00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 01000: bpp16 5-6-5 {R[4:0], G[5:0], B[4:0]} 01001: RESERVED 01010: RESERVED 01011: RESERVED 01100: bpp16 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]} 01101: RESERVED 01110: R8G8B8A8 01111: B8G8R8A8 1xxxx: RESERVED 0 = U8Y8V8Y8_OB 1 = Y8U8Y8V8_OB 2 = Y8V8Y8U8_OB 3 = V8Y8U8Y8_OB

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>4 = U8Y8V8Y8_TC
5 = Y8U8Y8V8_TC
6 = Y8V8Y8U8_TC
7 = V8Y8U8Y8_TC
8 = B5G6R5
9 = RESERVED9
10 = RESERVED10
11 = RESERVED11
12 = B5G6R5BS
13 = RESERVED13
14 = R8G8B8A8
15 = B8G8R8A8
16 = RESERVED16
17 = RESERVED17
18 = RESERVED18
19 = RESERVED19
20 = RESERVED20
21 = RESERVED21
22 = RESERVED22
23 = RESERVED23
24 = RESERVED24
25 = RESERVED25
26 = RESERVED26
27 = RESERVED27
28 = RESERVED28
29 = RESERVED29
30 = RESERVED30
31 = RESERVED31</td></tr>
<tr><td>4:0</td><td>X</td><td>SIFMT: This parameter defines the data format of source input. There are two groups of data formats, RGB format and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement.
00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format
00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement
01000: B5G6R5 5-6-5 {R[4:0], G[5:0], B[4:0]}
01001: RESERVED
01010: RESERVED
01011: RESERVED
01100: B5G6R5 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]}
01101: RESERVED
01110: R8G8B8A8
01111: B8G8R8A8
1xxxx: RESERVED
StretchBlit Inputs: RGB inputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV420 input = {YUV420 is converted into 4:2:2 UYVY via memory client} YUV422 inputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}
StretchBlit Outputs: RGB outputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV422 outputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}
StretchBlit Input/Output Rules:
<table>
<tr><th>src format</th><th>internal sb format</th><th>dst format</th></tr>
<tr><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td><td>R8G8B8A8</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*,</td><td>YUV 4:4:4</td><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*</td></tr>
</table></td></tr>
</table>

Bit	Reset	Description
		<table><tr><td>V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td></td><td></td></tr><tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr></table> StretchBlit color space converter supports YUV->RGB, YUV->YUV (gain) and RGB->RGB (gain). There is no support for RGB->YUV. 0 = U8Y8V8Y8_OB 1 = Y8U8Y8V8_OB 2 = Y8V8Y8U8_OB 3 = V8Y8U8Y8_OB 4 = U8Y8V8Y8_TC 5 = Y8U8Y8V8_TC 6 = Y8V8Y8U8_TC 7 = V8Y8U8Y8_TC 8 = B5G6R5 9 = RESERVED9 10 = RESERVED10 11 = RESERVED11 12 = B5G6R5BS 13 = RESERVED13 14 = R8G8B8A8 15 = B8G8R8A8 16 = RESERVED16 17 = RESERVED17 18 = RESERVED18 19 = RESERVED19 20 = RESERVED20 21 = RESERVED21 22 = RESERVED22 23 = RESERVED23 24 = RESERVED24 25 = RESERVED25 26 = RESERVED26 27 = RESERVED27 28 = RESERVED28 29 = RESERVED29 30 = RESERVED30 31 = RESERVED31

14.8.24 G2SB_CTX6_G2CONTROLSB_0

Offset: 0x601d │ Byte Offset: 0x18074 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000xxx)

Bit	Reset	Description
31	0x0	DISDW: Output destination writes go either to image memory or the EPP. 0: Output data is sent to memory 1: YUV or RGB data is passed directly to the EPP module, and no destination writes take place. 0 = GOTO_IMAGE_BUFFER 1 = GOTO_EPP
30	0x0	ENDITH: Enable Dithering (ENDITH). For 16 bit RGB output modes, the LSB of the color components can be modified by adding a variable residual value that will reduce the banding artifacts that can appear on the display. 0: Normal operation 1: Enable Dithering 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	0x0	KPOL: Key Signal Polarity (KPOL) Color/Chroma key signal is generated by comparing source input pixel color to a range of a color specified by lower and upper limit values. The key signal is interpreted in two ways, depending on which one of video and graphics images is foreground (and the other is background). This is effective only if Key signal generator is enabled. (see G2CMKEYL, G2CMKEYU) 0: Key signal is set to 1 when source pixel is within the lower and upper limit color range. 1: Key signal is set to 1 when source pixel is outside the lower and upper limit color range. 0 = WITHIN_BOUNDS 1 = OUTSIDE_BOUNDS
27	0x0	KEYEN: Key Signal Generator Enable (KEYEN). Key signal generator generates either chroma key signal (from YCbCr signal) or color key signal (from RGB signal) 0: Key signal generator is disabled. 1: Key signal generator is enabled. 0 = DISABLE 1 = ENABLE
26	0x0	DBSEL: StretchBLT Destination Buffer Selection (DBSEL). StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the destination video area. This bit selects one of the two buffering blocks to which this StretchBLT command delivers the destination image. The two buffering memory blocks are called A and B. 0: Destination image goes to ‘A’ buffer. 1: Destination image goes to ‘B’ buffer. 0 = DST_A 1 = DST_B
25	0x0	SBSEL: StretchBLT Source Buffer Selection (SBSEL). StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the source video area. This bit selects one of the two buffering blocks from which this StretchBLT command receives the source image. The two buffering memory blocks are called source-A and source-B. 0: Source image comes from ‘source-A’ buffer. 1: Source image comes from ‘source-B’ buffer. 0 = SRC_A 1 = SRC_B
24	0x0	SITYPE: StretchBLT Source Type (SITYPE). This bit identifies two types of source images. If source image is 2-to-1 interlaced and StretchBLT processes either one of the two interlaced field-images, physical (positional) displacement between the two interlaced fields must be taken into account. One field-image that is placed higher in position than the other field-image is called top-field and the other is called bottom-field. StretchBLT processing has to lower the top-field (or raise the bottom-field) to match the corresponding two target images in the overlay window (that is progressively scanned) right at the same position. If source image is full frame-image obtained from two interlaced field-images, its type is ‘top-field’. If source images are progressively scanned, the type identification is not significant and they may be designated either one of the two types 0: Source image is ‘top-field’. 1: Source image is ‘bottom-field’. 0 = TOP_FIELD 1 = BOTTOM_FIELD

Bit	Reset	Description
23	0x0	RANGEREDFRM:In the VC-1 specification, when the value of RANGEREDFRM variable (1-bit) for a picture is equal to 1, the picture shall be scaled up according to the following equation:Y = clip(((Y-128)*2) + 128);Cb = clip(((Cb-128)*2) + 128);Cr = clip(((Cr-128)*2) + 128);The clip operator limits the output to [0, 255].The input is also limited to [0, 255]. In the VC-1 nomenclature, the output (Y, Cb, Cr) tuple corresponds to the 'decoded' picture. The input (Y, Cb, Cr) tuple corresponds to the 'reconstructed' picture. The above equations create the decoded picture while keeping the reconstructed picture intact.Only YUV _OB formats are supported with range reducation enabled, not the YUV _TC or RGB formats. The YUV data coming from the video frame buffers (reconstructed picture) in the SDRAM was scaled up using the above equations and given to the display. The value of 1-bit wide RANGEREDFRM variable was configured in the DVO per frame and the frame was scaled whenever RANGEREDFRM == 1. 0 = DISABLE 1 = ENABLE
22:20	0x0	HFTYPE: StretchBLT Horizontal Filter Mode (HFTYPE[2:0]) The six-tap horizontal interpolation filter can be operated in various operation modes. For the image expansion, it should be programmed as a pure 6-tap interpolator. For the image contraction, it can work as partly lowpass filter and partly interpolater with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of horizontal filtering. 000: Pure interpolation filter. 001, 010, 011, 100, 101, 110: mix of interpolation and low pass filters 111: DISABLE. 0 = INTERP 1 = LPF1 2 = LPF2 3 = LPF3 4 = LPF4 5 = LPF5 6 = LPF6 7 = DISABLE
19	0x0	DISCSC: enabled color space converter by default 0 = ENABLE 1 = DISABLE
18	0x0	VFEN: StretchBLT Vertical Filter Enable (VFEN) Vertical filter shall be disabled if source images come from host CPU via CPU Read-FIFO. (SISEL) Vertical filter may be disabled to save some memory bandwidth but this will likely result in degradation of image quality. This option may also be used in the case where source image is progressive scanning and there is no vertical scaling. 0: Vertical filter is disabled. 1: Vertical filter is enabled. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
17:16	0x0	VFTYPE: StretchBLT Vertical Filter Mode (VFTYPE[1:0]). This bit is effective when the Vertical Filter is enabled. (bit 18) The two-tap vertical interpolation filter can be operated in various modes. For the image expansion, it should be programmed as a pure 2-tap interpolator. For the image contraction, it can work as partly averager and partly interpolator with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of vertical filtering. 00: Pure interpolation filter. 01: 25% averager, 75% interpolator. 10: 50% averager, 50% interpolator. 11: 100% averager. 0 = INTERP 1 = AVG25_INTERP75 2 = AVG50_INTERP50 3 = AVG
9:8	X	UVST: 00= 1/2 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 2 bytes. 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 4 bytes. 11= use uvstride. Note: Use care when using a tiled surface, since 1/2 or 1/4 of luma stride may yield an invalid pitch (stride). 0 = UVS2X 1 = UVS1X 2 = UVS4X 3 = UVS_G2UVSTRIDE
7	0x0	ENAHF: enable horizontal alpha filtering if disabled, use the alpha value of third tap for output pixel U,V line stride in 4:2:0 Format. 0 = DISABLE 1 = ENABLE
6	0x0	ENAVF:enable alpha vertical filter 0 = DISABLE 1 = ENABLE
5	0x0	IMODE: Source (input) Data Mode (IMODE). This parameter defines the data mode of source input. 0: Multiplexed Mode (data format is specified by SIFMT[2:0]). One block of source data in the Image Buffer memory contains multiplexed component signals. 1: Planar mode, 4:2:0 YUV (SIFMT[2] specifies data format of U/V components). Three blocks of source data in the Image Buffer memory contain separate Y, U, and V component signals. For a circular buffer input to gr2d, the input format cannot be planar. It must be a multiplex, i.e., no YUV420 planar circular buffer input to gr2d. 0 = MULTIPLEX 1 = PLANAR
4	0x0	YUV422PLANAR: Source data is yuv422 planar valid when imode==PLANAR
3	0x0	YUV422ROTATION: YUV422 planar rotated.

14.8.25 G2SB_CTX6_G2CONTROLSECOND_0

Offset: 0x601e │ Byte Offset: 0x18078 │ Read/Write: R/W │ Reset: 0x00000000 (0bxx000000x00xxxx000xxxx000000000x)

Bit	Reset	Description
29	0x0	FR_READWAIT: Fast Rotate wait for reads. Enabling this bit forces FR to wait for the reads to be stored in the data return FIFO before sending writes out to the same block. Always set to ENABLE. This field is debug only.

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
28:26	0x0	FR_TYPE: Fast Rotate type 000 = FLIP_X 001 = FLIP_Y 010 = TRANS_LR (mirrors about diagonal. Diagonal runs from upper left to lower right) 011 = TRANS_RL (mirrors about diagonal. Diagonal runs from upper right to lower left) 100 = ROT_90 (counterclockwise by 90 degrees) 101 = ROT_180 110 = ROT_270 (clockwise by 90 degrees) 111 = IDENTITY 0 = FLIP_X 1 = FLIP_Y 2 = TRANS_LR 3 = TRANS_RL 4 = ROT_90 5 = ROT_180 6 = ROT_270 7 = IDENTITY
25:24	0x0	FR_MODE: Fast Rotate mode sel. G2 Fast Rotate Transforms a surface via FR_TYPE transformation. Works in either 2-buffer (copy) or 1-buffer mode (in place) mode. The engine breaks down a larger surface into a grid of smaller FR_BLOCKs. Works on the granularity of an FR_BLOCK. An FR_BLOCK is: 16x16 pixel block (DSTCD = bpp8) 8x8 pixel block (DSTCD = bpp16) 4x4 pixel block (DSTCD = bpp32) Maximum surface size is 4096x4096 Key information - source and destination base address must be 128-bit word aligned - engine works on FR_BLOCK granularity: transformed surface width in multiples of 16 bytes** transformed surface height in multiples of 16/8/4 lines for bpp8/bpp16/bpp32 FR_BLOCK if surface dimension is not a multiple, software can program FR engine to transform larger surface (round up to next FR_BLOCK in width and height-->transform-->calculate relative memory pointer address) - during a rotational transformation (TRANS_LR, TRANS_RL, ROT_90, ROT_270): the stride of the output surface != the stride of the input surface when working on non-square input input: output: *^---------- *$---------- $ 0 1 2 3 4 5 6 7 ^ 24 16 8 0 ^ ^ ^ ^ | 8 9 10 11 12 13 14 15 | 25 17 9 1 ^ ^ ^ ^ |16 17 18 19 20 21 22 23 | 26 18 10 2 ^ ^ ^ ^ |24 25 26 27 28 29 30 31 | 27 19 11 3 ^ ^ ^ ^ | - - - - * * * * | 28 20 12 4 * * * * | - - - - * * * * | 29 21 13 5 * * * * | - - - - * * * * | 30 22 14 6 * * * * | - - - - * * * * | 31 23 15 7 * * * * - Tiling alignment restrictions subsume FR restrictions because the FR surface base address is the same as the start address. **Register Programming:** FR_MODE - inplace or copy FR_TYPE - type of transformation DSTCD - bpp8, bpp16, bpp32 SRCBA - source base address SRCWIDTH - (width in pixels-1) SRCHEIGHT - (height in lines-1) SRCS - source stride DSTBA - destination base address DSTS - destination stride FR_READWAIT - always set to enable FR inefficency in the following setup: 1. FR_MODE==SQUARE 2. if(FR_TYPE==YFLIP && SRCHEIGHT==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 Or if(FR_TYPE==XFLIP && SRCWIDTH==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 3. Then, the middle 8 lines/pixels, e.g., YFLIP/16bpp/n==3, line 8 to line 15 will be processed twice.

Bit	Reset	Description
		00 = disable fast rotate - this turns off the second level clock to fr_rotate engine. Please remember to do so! 01 = src/dst copy mode - two separate buffers 10 = square in place - one buffer 11 = blank 0 = DISABLE 1 = SRC_DST_COPY 2 = SQUARE 3 = BLANK
22:21	0x0	CLIPC: Clipping rectangle control, if clip enable, bit 57, XYTDW, should be cleared. 0x=clipping disabled 10=draw only inside clipping rectangle 11=draw only outside clipping rectangle
9	0x0	ALPSRCORDST: 32-bit blending mode, output alpha selection 0:source alpha, 1:destination alpha 0 = DISABLE 1 = ENABLE
8:4	0x0	ALPTYPE: Alpha blending method FIX: ALPHA blending: Fixed alpha, ALPHA is the value, B5G6R5 * B5G6R5 VCAA: B5G6R5->B5G6R5 PL1BPP: ALPHA blending: Alpha 1 bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero. PL2BPP: ALPHA blending: Alpha 2 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero. PL4BPP: ALPHA blending: Alpha 4 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero. PL8BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1 bit from source B5G5R5A1, alpha(MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4 bits from source A4B4G4R4, alpha(LSB). Dest: B5G6R5. PLS4BPP: ALPHA blending: Alpha 4 bits from source B4G4R4A4, alpha(MSB). Dest: B5G6R5. VCAA: R8G8B8A8->B5G6R5 without reading VCAA plane (surface blit with bpp down convert - implemented in hardware by the VCAA engine; not really a VCAA resolve) PLS8BPP: ALPHA blending: Alpha 8 bits from source/destination (decided by ALPSRCORDST), R8G8B8A8, alpha(MSB). DST: R8G8B8A8 VCAA: R8G8B8A8->R8G8B8A8, alpha has same blending method as RGB PLS8BX: ALPHA blending: Alpha 8 bits from source B8G8R8A8, alpha (MSB). Dest: B5G6R5. (**Restrictions) VCAA: R8G8B8A8->B5G6R5 PLS1BPPAL: ALPHA blending: Alpha 1 bit from source A1B5G5R5, alpha (LSB). Dest: B5G6R5 VCAA: A1B5G5R5->A1B5G5R5 **Restriction PLS8BX alpha blending has the following restrictions: 1. Source/destination addresses have to be in a 128-bit boundary. 2. Destination width has to be a multiple of 4 pixels. 3. Source/Destination strides have to be a multiple of 128 bits. 0 = FIX 1 = PL1BPP 2 = PL2BPP 3 = PL4BPP 4 = PL8BPP 5 = PL44BPP 6 = PLS1BPP 7 = PLS4BPPAL 8 = PLS4BPP 9 = PLS8BPP 10 = PLS8BX 11 = PLS1BPPAL

Bit	Reset	Description
3	0x0	BEWSWAP: Host port word swap 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 0 = DISABLE 1 = ENABLE

14.8.26 G2SB_CTX6_G2CONTROLMAIN_0

G2 Blit Formats:

(1) G2 Blit size is based on height in lines, width in pixels, stride in bytes, and pixel size (color depth). Unless alpha blending is enabled, G2 does not care about pixel components.

Source color depths supported by G2 Blit:

- color depth same as destination: G2CONTROLMAIN.SRCCD = 1
- monochrome: G2CONTROLMAIN.SRCCD = 0

(source monochrome color determined by G2SRCBGC, G2SRCFGC)

Pattern color depths supported by G2 Blit:

- color depth same as destination: G2PATOS.PATCD = 1
- monochrome: G2PATOS.PATCD = 0

(pattern monochrome color determined by G2PATBGC, G2PATFGC)

Destination color depths supported by G2 Blit:

- 1 byte per pixel: G2CONTROLMAIN.DSTCD = BPP8
- 2 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP16
- 4 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP32

(2) Alpha blending operates on 3 or 4 component pixels of a specified color depth. One of the components may be alpha (A), depending on the alpha blend format, which is only allowed in certain component positions defined below. The only format rule of the remaining components is that they must be in the same source and destination position because the alpha blend engine does not support component swapping. For example, valid 32BPP formats are: RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888, GRBA_8888->GRBA_8888, etc.

Source formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16:
  - ALPTYPE_PLS1BPP: xxxA_5551 (for example, BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
  - ALPTYPE_PLS4BPP: xxxA_4444 (for example, BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
  - (other) : xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32:
  - (all) : xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

Destination formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16: xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

src cd	src format	dst cd	dst format	examples
BPP8	no support	BPP8	no support	
BPP16	xyzA_5551, xyzA_4444, xyz_565	BPP16	xyz_565	BGRA_5551->BGR_565, RGB_565->RGB565
BPP32	xyzA_8888	BPP32	xyzA_8888	RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888
BPP32	xyzA_8888	BPP16	xyz_565	RGBA_8888->RGB_565, BGRA_8888->BGR_565

VCAA Resolve Engine

(1) Format support

srcColor format	srcCoverage format	legal dest format(s)
A1B5G5R5	VCAA	A1B5G5R5
R8G8B8A8	VCAA	R8G8B8A8, B5G6R5
B5G6R5	VCAA	B5G6R5

To program resolve type:

- ALPTYPE == FIX: B5G6R5 -> B5G6R5
- ALPTYPE == PLS1BPPAL: A1B5G5R5 -> A1B5G5R5
- ALPTYPE == PLS8BPP: R8G8B8A8 -> R8G8B8A8
- ALPTYPE == PLS8BX: R8G8B8A8 -> B5G6R5

(2) Color surface programming

- color surface base address: SRCBA
- subsurface start x: SRCX (pixel index)
- subsurface start y: SRCY (line index)
- color surface stride: SRCS (bytes)

The engine supports subsurface resolve. One can imagine a full color buffer of 1024x768, but

```
[pseudo_code]
      color_surface_start_address_fetch = pcurctx->regs.rG2SRCBA.uSRCBA() +
       (pcurctx->regs.rG2SRCPS.uSRCX() * VCAAState.color surface depth) +
       (pcurctx->regs.rG2SRCPS.uSRCY() * VCAAState.color_surface_stride);
```

(3) VCAA surface programming

- VCAA surface base address: PATBA
- VCAA surface stride: PATST (bytes)

```
[pseudo_code]
      vcaa_surface_start_address_fetch = pcurctx->regs.rG2PATBA.uPATBA();
```

Software *MUST* directly program this register to the proper sub-surface location in the vcaa surface which corresponds to the SRCX and SRCY programming. The calculation is PATBA = vcaa_base_address + SRCX * (1) + SRCY * PATST;

(4) Resolve surface programming

- resolve window width: DSTWIDTH (pixels)

- resolve window height: DSTHEIGHT (lines)
- resolve output stride: DSTS (bytes)

These values program the width and height of the resolved surface. In the base case:

- DSTWIDTH = color surface width
- DSTHEIGHT = color surface height
- DSTS = resolve surface format bpp * DSTWIDTH

The VCAA engine technically supports DSTX and DSTY (like SRCX and SRCY).

```
[pseudo_code]
   resolve_surface_start_address_put = pcurctx->regs.rG2DSTBA.uDSTBA() +
     (pcurctx->regs.rG2DSTPS.uDSTX() * resolve_surface_color_depth) +
     (pcurctx->regs.rG2DSTPS.uDSTY() * resolve_surface_color_stride);
```

(5) Resolving a pixel

The coverage surface is an 8 bits per pixel surface (C4X4). The surface is initialized to all 1s by 3d.

For a given pixel, we need to look up 4 bits of coverage data. If the data is 0xf, the pixel does not need to be resolved. Otherwise, the pixel needs to be reblended with certain weightings of its neighbors (i.e., we are on the edge of some geometry).

If any coverage bits are 0, the color needs to be resolved. To do so, calculate the following equation:

```
color_new = (
              20*color_old +
              27*(
                  ((cover_down  == 0) ? color_down  : color_old) +
                  ((cover_right == 0) ? color_right : color_old) +
                  ((cover_left  == 0) ? color_left  : color_old) +
                  ((cover_up    == 0) ? color_up    : color_old)
                  )
            ) / 128;
```

If coverage bit == 0, then use the neighbor color value in the resolve

If coverage bit == 1, then use the center color value in the resolve

The resolve is done on a per channel (r/g/b/a) basis:

format convert input channel to 8-bit

perform resolve equation above

format convert to output format

For the resolution of the alpha channel:

- In B5G6R5: No alpha information. Do nothing.
- In A1B5G5R5: No resolve. Keep the source color alpha bit.
- In R8G8B8A8: Normal 8-bit channel resolve

(6) Resolving a pixel on the edge

When on an edge, the VCAA engine treats the coverage bit as a 1. Use the center color value. This holds true even on the edge of the sub-surface resolve window when located in the middle of the larger surface

(7) Maximum dimensions

The maximum resolve subsurface is 4096 pixels wide.

(8) Surface restrictions

The VCAA engine has the following restrictions with respect to input surfaces

- input base address is 128-bit memory word aligned
- input stride is a multiple of 128 bits (16 bytes)

The VCAA engine has the following restrictions with respect to output surfaces

- output base address is 128-bit memory word aligned
- output stride is a multiple of 128 bits (16 bytes)

(9) Coverage surface layout

The surface is C4X4.

Surface layout (C4X4)

```
(lsb2msb)  C0_C1_C2_C3_X_X_X_X     (X is a don't care.  Qrast inits to 1)
(msb2lsb)  X_X_X_X_C3_C2_C1_C0



     +          o          o
          C0
             C1   Cov(0,1)
        C2
          C3
     o          o          o

       Cov(1,0)   Cov(1,1)

     o          o          o
     +/o = pixel centers
```

In the example above,

```
        Cov(0,0) = X_X_X_X_C3_C2_C1_C0
        C3       = V_C3(Cov(0,0));
        ..         ..
        C0       = V_C0(Cov(0,0));
     #define V_C3(bits) (bits & 0x8)
     #define V_C2(bits) (bits & 0x4)
     #define V_C1(bits) (bits & 0x2)
     #define V_C0(bits) (bits & 0x1)
```

Resolving a pixel

In an ideal world, the coverage would be simply laid out such that when resolving Pixel(x,y), we would need only to fetch Cov(x,y) to get all 4 bits of information.

In actuality, the coverage surface was rearranged to help qrast in performance. To resolve Pixel(x,y) gr2d has to fetch 4 C4X4 bytes [Cov(x,y), Cov(x-1,y), Cov(x,y-1), Cov(x-1,y-1)]. From each C4X4 byte, it extracts one bit which is used in the resolve equations shown in (5).

Need the logical VCAA bits (DRLU) for blend which is spread across four C4X4 coverage bytes

```
          U

    L

        o

            R

      D
```

Where:

o = Pixel(x,y)

D = V_C1(Cov(x-1,y))

R = V_C0(Cov(x,y))

L = V_C3(Cov(x-1,y-1))

U = V_C2(Cov(x,y-1))

Offset: 0x601f | Byte Offset: 0x1807c | Read/Write: R/W | Reset: 0x00000000 (0bx000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: Pattern Data Select: SRCSEL and PATSEL cannot be both enabled. 0 = SCREEN 1 = MEMORY
28	0x0	DSTDIR: Destination direct addressing
27	0x0	SRCDIR: Source direct addressing
26:25	0x0	GCSW: Display Switching Window Control (GCSW[1:0]). This parameter controls multi-buffering for Display. x0: At end of current command, do not send signal to Display to switch buffer. 01: two buffers, dstba and dstba_b are used 11: three buffers, dstba, dstba_b, dstba_c are used
24	X	SRCBAS: Source base address select: 0=srcba, 1=dstba. This is not used for Line Draw and if source data comes from host memory.
23:22	0x0	SRCT: Source transparency enable: 0x=source transparency disabled, 10=mono source background transparency or color source transparency, 11=mono source foreground transparency or inverse color source transparency. NOTE: When source transparency is enabled and SRCCD==0(mono) SRCBGC!=SRCFGC should be satisfied.
21	X	HLMONO: Start from Msb or lsb in byte when mono expansion If HLMONO is 1, bit 0 (the lsb) is the first bit. If HLMONO is 0, bit 7 (the msb) is the first bit
20	X	SRCCD: 0 Source mono 1 Source has same color depth as destination SRCCD==0 xdir/ydir has to be 0

Bit	Reset	Description
19:18	0x0	DSTT: Destination read transparency enable: 0x=destination read transparency disabled 10=color destination read transparency 11=inverse color destination read transparency. 20 rw INIC Initiate Command (1=initiate command immediately, 0=wait for launch write)
17:16	X	DSTCD: Destination color depth: 00=8-bpp, 01=16-bpp, 10=32-bpp. 11=reserved 0 = BPP8 1 = BPP16 2 = BPP32 3 = RESERVED3
15	0x0	SRCSEL: Source Data Select: 0 = SCREEN 1 = MEMORY
14	0x0	YFLIP: flip y direction to make image upside down or the other way. If YFLIP==1, ROP cannot include destination. 0 = DSIABLE 1 = ENABLE
13	0x0	PATPACK: Pattern Data is in Pack Mode. PATLNGAP in G2PATPACK is the line gap for pattern packed mode If (PATPACK && ~PATSEl), pattern data is packed and from screen, PATMONOW/PATMONOH should be programmed properly to fetch pattern data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
12	0x0	SRCPACK: Source Data is in Pack Mode. SRCLNGAP in G2SRCPACK is the line gap for source packed mode. If (SRCPACK && ~SRCSEL), source data is packed and from screen, SRCMONOW/SRCMONOH should be programmed properly to fetch data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
11	0x0	XYTDW: xy transpose, Line stride DSTS has to be 16-byte aligned if enabled. If XYTDW==1, ROP cannot include destination.
10	0x0	YDIR: 0=incrementing, 1=decrementing. ydir should not be set when source surface has different color depth as destination surface. SRCCD==0 (mono src) OR PATCD==0 (mono pat), ydir has to be 0
9	0x0	XDIR: 0=incrementing, 1=decrementing. xdir should not be set when source surface has different color depth as destination surface. For example, ALPTYPE=PLS8BX (32bpp blending with 16bpp), xdir has to be 0 SRCCD==0 (mono src) OR PATCD==0 (mono pat), xdir has to be 0
8	0x0	PATFL: When mono pattern is set, we use mono tile pattern fill. current pattern client can support a 16x16 bit tile which can be used to generate a larger surface (i.e., tile this surface in the x and y direction) The tile is stored in memory. Currently, PATXO and PATYO indicate where to start in the 16x16 tile when expanding the monochrome data. See patxo and patyo comments below for programming When xdir==1 Patxo = pattern width - (patxo+destination width)&0xF -- For xoffsets y offsets PATYO remains the same when xdir/ydir change. How the tile replication pattern looks based on xdir/ydir (without programming patxo above): xdir=0 ydir=0 xdir=1 ydir=0 +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ // xdir=0 ydir=1 xdir=1 ydir=1 Mono tile is +---+ |^|^| |~~~| +---+

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
7	0x0	PATSLD: BitBlt Solid Pattern Fill: 1=enable. BGC will be used as the color value. 0 = DISABLE 1 = ENABLE
6	0x0	SRCSLD: BitBlt Solid Source Color Fill: 1=enable. FGC will be used as the color value. 0 = DISABLE 1 = ENABLE
5	0x0	ALPEN: BitBlt Alpha Blending, 1=enable. 0=disable,when both Faden and alpen are 1, output=Source*alpha_v + fadoff, alpha_v is decided by alptype
4	0x0	FADEN: BitBlt Source Copy Fade enable, 1=enable (share with mltln), only support 16bpp mode 0 = DISABLE 1 = ENABLE
3	0x0	TEST0BIT: Command finish timing bit 0: 2D command finishes when last data has been pushed to memory write client. 1: 2D command waits memory write client to be idle to finish. 0 = DISABLE 1 = ENABLE
2	0x0	TURBOFILL: fast fill rectangle in 128 bits/clock. Some limitations with this mode:srcsld==1 rop==0xcc, no clipping, no transparency, xdir==0, ydir==0, flip==0, xytdw==0. Results are undefined if the above limitations are not satisfied.
1:0	0x0	CMDT: Command Type: 00=BitBlt 01=Line Draw 10=VCAA 11=reserved When the raise command is executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

14.8.27 G2SB_CTX6_G2ROPFADE_0

Offset: 0x6020 | Byte Offset: 0x18080 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:24	X	FADOFF:
23:16	X	FADCOE:
7:0	0x0	ROP:If YFLIP==1 or XYTDW==1, ROP cannot include destination.Since destination may have been corrupted before reading out.

14.8.28 G2SB_CTX6_G2ALPHABLEND_0

Offset: 0x6021 | Byte Offset: 0x18084 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
31:24	X	ALPHA1V

Bit	Reset	Description
23:16	X	ALPHA0V
8	0x0	ALPHAINV
7:0	X	ALPHA

14.8.29 G2SB_CTX6_G2CLIPLEFTTOP_0

ClipRect Methods. top-left bounds are inclusive

Offset: 0x6022 | Byte Offset: 0x18088 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPT
14:0	X	CLIPL

14.8.30 G2SB_CTX6_G2CLIPRIGHTBOT_0

Bottom-right bounds are exclusive

Offset: 0x6023 | Byte Offset: 0x1808c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPB
14:0	X	CLIPR

14.8.31 G2SB_CTX6_G2PATPACK_0

Pattern Packed Mode

Pattern Methods

G2PATPACK should be used to specify the line gap. Use G2PATPACK_SIZE to program height and width. PACK is generally only useful with narrow monochrome surfaces.

Note: PACK is not officially supported when the surface is tiled.

Offset: 0x6024 | Byte Offset: 0x18090 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

14.8.32 G2SB_CTX6_G2PATPACK_SIZE_0

Pattern Packed Mode

Packed mode, pattern data line gap. Byte

G2PATPACK_SIZE -- extension of G2PATPACK, this register only holds the size of the packed data

Note: PACK is not officially supported when the surface is tiled.

Offset: 0x6025 | Byte Offset: 0x18094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	PATMONOH: Pattern mono data height in packed mode, byte
15:0	X	PATMONOW: Mono data width in packed mode, byte

14.8.33 G2SB_CTX6_G2PATBA_0

Offset: 0x6026 | Byte Offset: 0x18098 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBA: If (PATFL==1){ It has to be 16-byte aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

14.8.34 G2SB_CTX6_G2PATOS_0

Offset: 0x6027 | Byte Offset: 0x1809c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00xxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	PATYO: y offset for mono tile pattern fill. see PATFL
27:24	X	PATXO: x offset for mono tile pattern fill. see PATFL
22:21	0x0	PATT: Mono pattern transparency enable: 0x=pattern transparency disabled, 10=mono pattern background transparency or color pattern transparency, 11=mono pattern foreground transparency or inverse color pattern transparency. NOTE: When pattern transparency is enabled and PATCD==0(mono) PATBGC!=PATFGC should be satisfied.
16	X	PATCD:0 mono1 same as dstcd PATCD==0 xdir/ydir has to be 0
15:0	X	PATST:stride

14.8.35 G2SB_CTX6_G2PATBGC_0

Offset: 0x6028 | Byte Offset: 0x180a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

14.8.36 G2SB_CTX6_G2PATFGC_0

Offset: 0x6029 | Byte Offset: 0x180a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

14.8.37 G2SB_CTX6_G2PATKEY_0

Offset: 0x602a | Byte Offset: 0x180a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

14.8.38 G2SB_CTX6_G2DSTBA_0

Offset: 0x602b | Byte Offset: 0x180ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA: Destination base address (byte address)

14.8.39 G2SB_CTX6_G2DSTBA_B_0

Offset: 0x602c | Byte Offset: 0x180b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_B: Destination base address (byte address) only usable in hardware trigger mode by enable gcsw

14.8.40 G2SB_CTX6_G2DSTBA_C_0

Offset: 0x602d | Byte Offset: 0x180b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_C: Destination base address (byte address)

14.8.41 G2SB_CTX6_G2DSTST_0

Offset: 0x602e | Byte Offset: 0x180b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

14.8.42 G2SB_CTX6_G2SRCPACK_0

source data packed mode

Surface Methods

G2SRCPACK should only be used to specify the line gap. Use G2SRCPACK_SIZE to program height and width.

Offset: 0x602f | Byte Offset: 0x180bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

14.8.43 G2SB_CTX6_G2SRCPACK_SIZE_0

Source data packed mode

In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, the required width/stride is 16 byte aligned. Packed mode - source mono data line gap

G2SRCPACKS_SIZE -- extension of G2SRCPACK, this register only holds the size of the packed data

Offset: 0x6030 │ Byte Offset: 0x180c0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCMONOH: Packed mode - source mono data height in bytes
15:0	X	SRCMONOW: Packed mode - source mono data width in bytes

14.8.44 G2SB_CTX6_G2SRCBA_0

src base address

Offset: 0x6031 │ Byte Offset: 0x180c4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA: Source base address (byte address)

14.8.45 G2SB_CTX6_G2SRCBA_B_0

SB only

Offset: 0x6032 │ Byte Offset: 0x180c8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA_B: This parameter specifies the start address of source image stored in the image buffer memory. In 4:2:0 format mode, this image block accommodates for Y-image.This address specifies byte position, however, bits [2:0] are restricted with respect to the data formats to fit multiple pixels in one memory word (8 bytes). For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format. Since one Y pixel takes 8 bits, all 8 byte positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there are no restrictions on this value.)

14.8.46 G2SB_CTX6_G2SRCST_0

Offset: 0x6033 │ Byte Offset: 0x180cc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS: Source Stride coordinate (bytes) with respect to SRCBA.In order to fit multiple pixels in one memory word (8 bytes), bits [2:0] are restricted with respect to the data formats. For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format.

14.8.47 G2SB_CTX6_G2SRCBGC_0

srcColors

Offset: 0x6034 | Byte Offset: 0x180d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

14.8.48 G2SB_CTX6_G2SRCFGC_0

Offset: 0x6035 | Byte Offset: 0x180d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

14.8.49 G2SB_CTX6_G2SRCKEY_0

Offset: 0x6036 | Byte Offset: 0x180d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

14.8.50 G2SB_CTX6_G2SRCSIZE_0

Offset: 0x6037 | Byte Offset: 0x180dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	SRCHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	SRCWIDTH

14.8.51 G2SB_CTX6_G2DSTSIZE_0

Offset: 0x6038 | Byte Offset: 0x180e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	DSTHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	DSTWIDTH: In 2D mode, the largest number can be programmed is 0x7fef

14.8.52 G2SB_CTX6_G2SRCPS_0

ImageBlit Methods

Offset: 0x6039 | Byte Offset: 0x180e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCY
15:0	X	SRCX:SRCX[2:0] are ignored in SRCCD==0 (mono expansion). The first bit of the first byte (bit 7 if HLMONO==0, or bit 0 if HLMONO==1) always expand to DSTX,DSTY

14.8.53 G2SB_CTX6_G2DSTPS_0

Offset: 0x603a | Byte Offset: 0x180e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	DSTY
15:0	X	DSTX: NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero

14.8.54 G2SB_CTX6_G2CBDES_0

Circular Buffer

Offset: 0x603b | Byte Offset: 0x180ec | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TOPCLIP: Top clipping at the first buffer (buffer start), refer to TOP_CBLINE in G2CBDES2 0 = DISABLE 1 = ENABLE
30:16	X	CBLINE: Vertical line number in one buffer
7:0	X	CBCOUNT: This specifies the number of buffers incircular buffer feature

14.8.55 G2SB_CTX6_G2CBSTRIDE_0

Offset: 0x603c | Byte Offset: 0x180f0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	X	CBUVSTRIDE: Chroma Buffer Stride default is half of luma 00= 1/2 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 2 bytes. 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 4 bytes. 1x= Reserved 0 = CBS2X 1 = CBS1X 2 = CBS4X
23:0	X	CBSTRIDE: Video Buffer Luma (or RGB) Buffer Stride. This is luma buffer stride (in bytes)

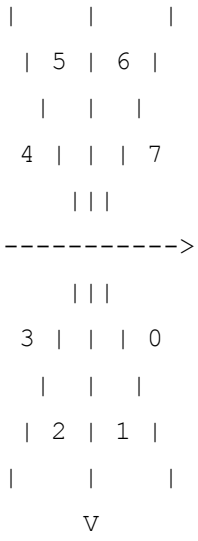
14.8.56 G2SB_CTX6_G2LINESETTING_0

Line Methods

Offset: 0x603d | Byte Offset: 0x180f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	OCTANTS: 000: octant 0 001: octant 1 010: octant 2 011: octant 3 100: octant 4 101: octant 5 110: octant 6 111: octant 7
28	X	LINEUSEOCTANT:use OCTANTS in G2LINEDELTAN register instead of MAJOR LINEXDIR LINEYDIR
27	X	DROPLASTP:draw last pixel or not
26	X	LINEYDIR
25	X	LINEXDIR
24	X	MAJOR: 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

14.8.57 G2SB_CTX6_G2LINEDELTAN_0



Offset: 0x603e | Byte Offset: 0x180f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

14.8.58 G2SB_CTX6_G2LINEDELTAM_0

Offset: 0x603f | Byte Offset: 0x180fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

14.8.59 G2SB_CTX6_G2LINEPOS_0

Offset: 0x6040 | Byte Offset: 0x18100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LINEYPOS
15:0	X	LINEXPOS

14.8.60 G2SB_CTX6_G2LINELEN_0

Offset: 0x6041 | Byte Offset: 0x18104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

14.8.61 G2SB_CTX6_G2CSCFOURTH_0

Offset: 0x6042 | Byte Offset: 0x18108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:16	X	G2V: Multiplier for G for V generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.368 (decimal) or 0x12F. For any other combination, this parameter is ignored.
8:0	X	G2U: Multiplier for G for U generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.291 (decimal) or 0x125. For any other combination, this parameter is ignored.

14.8.62 G2SB_CTX6_G2SRCST_B_0

Offset: 0x6043 | Byte Offset: 0x1810c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

14.8.63 G2SB_CTX6_G2UVSTRIDE_0

Offset: 0x6044 | Byte Offset: 0x18110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

14.8.64 G2SB_CTX6_G2CBDES2_0

Circular Buffer Controller 2

Offset: 0x6045 | Byte Offset: 0x18114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

14.8.65 G2SB_CTX6_G2TILEMODE_0

Offset: 0x6046 | Byte Offset: 0x18118 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx0xxx0xxx0xxx0xxx0xxx0)

Bit	Reset	Description
20	0x0	DST_WR_TILE_MODE:destination surface 0 = LINEAR 1 = TILED
16	0x0	DST_RD_TILE_MODE:Same as destination write unless DST_RD_WR_SEP (not supported) 0 = LINEAR 1 = TILED
12	0x0	PAT_UV_TILE_MODE:UNUSED 0 = LINEAR 1 = TILED
8	0x0	PAT_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED
4	0x0	SRC_UV_TILE_MODE:UV surface, ignored in RGB mode 0 = LINEAR 1 = TILED
0	0x0	SRC_Y_TILE_MODE:Y or RGB surface 0 = LINEAR 1 = TILED

14.8.66 G2SB_CTX6_G2PATBASE_0

Offset: 0x6047 | Byte Offset: 0x1811c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PAT_BASE:pattern base address in tile mode, PATBA is the linear address where pixel start

14.8.67 G2SB_CTX6_G2SRCBA_SB_SURFBASE_0

SB_SURFBASE Registers

These registers need only be programmed when using SB. They point to the base of the various source and destination surfaces. Technically they are only needed when tiling is enabled, but there is no harm in always programming them.

Their counterpart registers (_SB_SURFBASE with SB_SURFBASE stripped off) indicate the location of the first pixel to be sourced or written within the surface.

This register exists to mimic the X, Y (SRCX, SRCY) functionality found in the BitBlt engine.

For example, to get at pixel X, Y:

SRCBA = SRCBA_SB_SURFBASE + Y*stride + X*Bpp

Offset: 0x6048 | Byte Offset: 0x18120 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRC_ADDR: surface address corresponding to G2SRCBA: -base of interleaved sources (RGB, YUV) -base of Y plane. Only used by the StretchBlit Engine

14.8.68 G2SB_CTX6_G2DSTBA_SB_SURFBASE_0

Offset: 0x6049 | Byte Offset: 0x18124 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_ADDR: surface address corresponding to G2DSTBA. Only used by the StretchBlit Engine

14.8.69 G2SB_CTX6_G2DSTBA_B_SB_SURFBASE_0

Offset: 0x604a | Byte Offset: 0x18128 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_B_ADDR: surface address corresponding to G2DSTBA_B. Only used by the StretchBlit Engine, and G2CONTROLSB.DBSEL() is enabled

14.8.70 G2SB_CTX6_G2VBA_A_SB_SURFBASE_0

Offset: 0x604b | Byte Offset: 0x1812c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	V_ADDR: surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of V plane. Only used by the StretchBlit Engine

14.8.71 G2SB_CTX6_G2UBA_A_SB_SURFBASE_0

Offset: 0x604c | Byte Offset: 0x18130 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	U_ADDR: surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of U plane. Only used by the StretchBlit Engine

14.9 G2SB CTX 7 Registers

14.9.1 G2SB_CTX7_INCR_SYNCPT_0

Memory Controller Tiling Definitions

Offset: 0xa000 | Byte Offset: 0x28000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4

Bit	Reset	Description
		5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	INDX: syncpt index value

14.9.2 G2SB_CTX7_INCR_SYNCPT_CNTRL_0

Offset: 0xa001 │ Byte Offset: 0x28004 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

14.9.3 G2SB_CTX7_INCR_SYNCPT_ERROR_0

Offset: 0xa002 │ Byte Offset: 0x28008 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

14.9.4 G2SB_CTX7_G2CLASSCHANNEL_REGONLY_0

This register can be accessed by both register read/write and command FIFO in order not to break exsting code, keep suffix REGONLY.

2D syncpt behavior

- Each 2D context has its own syncpt client.
- OP_DONE increments when 2D's memory write client is idle, meaning all preceding operations are complete.
- RD_DONE behaves like OP_DONE
- REG_WR_SAFE behaves like IMMEDIATE meaning a syncpt increment returns immediately and does not indicate that a register write will NOT block.
- The implications are that 2D's host FIFO can fill and block access to 2D, which will indeterminately block a direct CPU access.

Offset: 0xa008 │ Byte Offset: 0x28020 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID
19:16	RW	X	CURR_CHANNEL

Bit	R/W	Reset	Description
9:0	RO	X	CURR_CLASS

14.9.5 G2SB_CTX7_G2TRIGGER_0

Control methods, since they are not programmed often, put them in the beginning

Offset: 0xa009 | Byte Offset: 0x28024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

14.9.6 G2SB_CTX7_G2TRIGGER1_0

Offset: 0xa00a | Byte Offset: 0x28028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

14.9.7 G2SB_CTX7_G2TRIGGER2_0

Offset: 0xa00b | Byte Offset: 0x2802c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

14.9.8 G2SB_CTX7_G2CMDSEL_0

Offset: 0xa00c | Byte Offset: 0x28030 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0xxx0xxxxxxxx00x0x0000000xxxx)

Bit	R/W	Reset	Description
28	RW	0x0	PRIORITY: indicates priority of the context, note vi-triggered contexts take priority over HIGH 0 = LOW 1 = HIGH
27:25	RW	X	LINKVAL
24	RW	0x0	LINKEN:Test purpose
23:16	RW	X	BUFFER_INDEX
15	RW	0x0	FRAME_END
14	RW	0x0	FRAME_START:host writes this bit to tell which buffer is ready
10	RW	0x0	CLIP_SOURCE_TOP_BOTTOM: Reserved for hardware test. Software should not use this bit. 0 = DISABLE 1 = ENABLE
9:8	RW	0x0	G2OUTPUT: 00: to memory. 01: to EPP 0 = MEMORY 1 = EPP 2 = RESERVED2 3 = RESERVED3
7	RW	0x0	CBSBDISABLE

Bit	R/W	Reset	Description
6	RW	0x0	HOSTTRIGGER:host trigger the command, host also need program circular buffer related data
5	RW	0x0	VITRIGGER:VI trigger enable, can be circular buffer or full frame
4	RW	0x0	CBENABLE:circular buffer feature enable 0= disable 0 = DISABLE 1 = ENABLE
0	RO	X	SBOR2D: 0 = G2 1 = SB

14.9.9 G2SB_CTX7_G2RAISE_0

Offset: 0xa00d │ Byte Offset: 0x28034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE: Push back to read FIFO when all commands in the channel are done.

14.9.10 G2SB_CTX7_G2HOSTSET_0

Offset: 0xa00f │ Byte Offset: 0x2803c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:4	0x0	HSTFPXL: specifies the first pixel location (in byte) in the first source data entry. In mono mode pixel should always start in byte boundary
3:0	0x0	HSTLNGAP: specifies distance (in byte) from last pixel of a line to the first pixel of next line. In mono mode, the value is the nearest byte number. e.g., if gap is 3 bits, HSTLNGAP should be 1 (byte), if gap is 9 bits, HSTLNGAP should be 2 (byte)

14.9.11 G2SB_CTX7_G2HOSTFIFO_0

Offset: 0xa010 │ Byte Offset: 0x28040 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

14.9.12 G2SB_CTX7_G2VDDA_0

Offset: 0xa011 │ Byte Offset: 0x28044 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VDSTEP: Vertical Scaling DDA. Reset value: xxxx-xxxxh. Vertical DDA Step (Increment) Value (VDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for vertical scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_height-1-VDTINI) / (Actual_destination_height-1). Truncate the remaining bits to keep the 12-bit fraction. Since we have to meet (Actual_source_height-1)*1.0 >= (Actual_destination_height - 1)*VDSTEP + VDTINI. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half line-spacing for up to approximately 2000 target lines.

Bit	Reset	Description
		For example, image expansion from 240 lines to 400 lines VDSTEP[17:0] = 19'b00_0000_1001_1001_1010 and image contraction from 240 lines to 150 lines VDSTEP[17:0] = 19'b00_0001_1001_1001_1010.

14.9.13 G2SB_CTX7_G2VDDAINI_0

Offset: 0xa012 | Byte Offset: 0x28048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:8	X	VDBINI: Vertical DDA Initial Value for Bottom-Field (VDBINI[7:0]) (see description above)
7:0	X	VDTINI: Vertical Scaling DDA Initial Values. Vertical DDA Initial Value for Top-Field (VDTINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for vertical scaling. Given a pair of consecutive source image lines representing positions 0.0 and 1.0, a destination line-image will be created at any position in [0.0, 1.0), that is specified by the fraction part of the vertical DDA. The 8-bit initial fraction value specifies the position of the first output (destination) line-image located between the first and the second input (source) line-images, respectively at position 0.0 and 1.0.This parameter provides a way to compensate relative displacement of top and bottom fields of a source image. Suppose, 2-to-1 interlaced original video field images #1 and #2 are in the image buffer, to be displayed in an overlay window. Assume that the #1 field image is the top-field. Using VDTINI = 8'hC0 for the top-field image and VDBINI = 8'h40 for the bottom-field image, for example, the two images from #1 and #2 fields get mapped to identically positioned destination images in the overlay window.

14.9.14 G2SB_CTX7_G2HDDA_0

Offset: 0xa013 | Byte Offset: 0x2804c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HDSTEP: Horizontal Scaling DDA. Reset value: xxxx-xxxxh. Horizontal DDA Step (Increment) Value (HDSTEP[18:0]) (upper 13 bits should be set to '0') This parameter specifies the increment value of the DDA used for horizontal scaling and it is in the form of 6-bit integer and 12-bit fraction. This value is determined by the equation: (Actual_source_width-1-HDINI) / (Actual_destination_width-1). Truncate the remaining bits to keep the 12-bit fraction. The 6-bit integer allows maximum contraction ratio of 1/64, and 12-bit fraction ensures the maximum limit of accumulated error less than half pixel-spacing for up to approximately 2000 destination pixels. For example, image expansion from 720 pixels to 800 pixels HDSTEP[17:0] = 19'b000_0000_1110_0110_0111 and image contraction from 720 pixels to 250 pixels HDSTEP[17:0] = 19'b000_0010_1110_0001_0101.

14.9.15 G2SB_CTX7_G2HDDAINILS_0

Offset: 0xa014 | Byte Offset: 0x28050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	HDINI: Horizontal Scaling DDA Initial Value. Horizontal DDA Initial Value (HDINI[7:0]) This parameter specifies the fraction part of initial value of the DDA used for horizontal scaling.Given a group of six consecutive source pixels that the two pixels at the center representing positions 0.0 and 1.0, a destination pixel will be created at any position in [0.0, 1.0), that is specified by the fraction part of the horizontal DDA. The 8-bit initial fraction value specifies the position of the first output (destination) pixel located between the first and the second input (source) pixels, respectively at position 0.0 and 1.0.For horizontal scaling this value may normally be set to 0.

14.9.16 G2SB_CTX7_G2CSCFIRST_0

The next 3 registers contain 8 parameters used by the YCbCr (YUV) to RGB color space conversion logic. This logic can also be used for RGB gain/gamma correction in RGB to RGB modes, and for luma gain in YUV to YUV modes.

Two new parameters (G2U, G2V) have been added for RGB to YUV conversion. These are in address 0x38. The equations have the form:

if (yuv2rgb)

(r,g,b) = matrix*(y+sbreg_yos,u,v)

matrix:	sbreg_cyx,	sbreg_cur,	sbreg_cvr,
	sbreg_cyx,	sbreg_cug,	sbreg_cvg,
	sbreg_cyx,	sbreg_cub,	sbreg_cvb,

else if (rgb2rgb)

(r,g,b) = matrix*(r,g+sbreg_yos,b)

matrix:	sbreg_cvr,	0,	sbreg_cur,
	sbreg_cvg,	sbreg_cyx,	sbreg_cug,
	sbreg_cvb,	0,	sbreg_cub,

else if (yuv2yuv)

(y,u,v) = matrix*(y,u,v)

matrix:	sbreg_cub,	0,	sbreg_cvb,
	sbreg_cug,	sbreg_cyx,	sbreg_cvg,
	sbreg_cur,	0,	sbreg_cvr,

else if (rgb2yuv)

(y,u,v) = matrix*(r,g,b) + (sbreg_yos,0,0)

matrix:	sbreg_cvb,	sbreg_cyx,	sbreg_cub,
	sbreg_cvg,	sbreg_g2u,	sbreg_cug,
	sbreg_cvr,	sbreg_g2v,	sbreg_cur,

Offset: 0xa015 │ Byte Offset: 0x28054 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	YOS:Y-Offset (YOFFSET[7:0]) for YUV generationThis parameter consists of 8-bit 2's complement in the range [-128,127]. For YUV->RGB, the recommended value is -16 (decimal) or 0xF0. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is +16 (decimal) or 0x10
21:12	X	CVR: multiplier for V/R for V or R generation.This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 1.5960 (decimal) or 0x0CCIf source data is in RGB format, this parameter may be used as gain adjustment for R component. For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038
9:0	X	CUB: multiplier for U/B for Y or B generation. Consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, the recommended value is 2.0172 (decimal) or 0x102. If source data is in RGB format, this parameter may be used as gain adjustment for B component. This register changes precision when doing RGB to YUV conversion (SIFMT=1xxx, DIFMT=0xxx). CUB becomes s1.8 and the recommended value is +0.098 or 0x019

14.9.17 G2SB_CTX7_G2CSCSECOND_0

Offset: 0xa016 | Byte Offset: 0x28058 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	CYX: Multiplier for Y/G (G gain). This positive-only parameter consists of 8-bit magnitude (1.7). For YUV->YUV, the recommended value is 1.1644 (decimal) or 0x95. For YUV->YUV or RGB->RGB, this may be used as gain adjustment for Y or G component. For RGB->YUV, the recommended value is +0.504 (decimal) or 0x041
21:12	X	CUR: Multiplier for U/B for V or R generation.This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This parameter takes non-zero value if hue is rotated. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.071 (decimal) or 0x209
8:0	X	CUG: Multiplier for U/B for U or G generation. Consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB, the recommended value is -0.3918 (decimal) or 0x132. For RGB->RGB, this parameter should be set to 0. For YUV->YUV, this parameter should be set to 1 (0x080). For RGB->YUV, the recommended value is +0.439 (decimal) or 0x038

14.9.18 G2SB_CTX7_G2CSCTHIRD_0

Offset: 0xa017 | Byte Offset: 0x2805c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	CVB: multiplier for V/R for Y or B generation. This parameter consists of a sign bit and 9-bit magnitude (s2.7). For YUV->RGB, normally this parameter is programmed to 0.0. This coefficient takes non-zero value if hue is rotated. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is +0.257 (decimal) or 0x021
8:0	X	CVG: multiplier for V/R for U or G generation.This parameter consists of a sign bit and 8-bit magnitude (s1.7). For YUV->RGB, the recommended value is -0.8130 (decimal) or 0x168. For YUV->YUV & RGB->RGB, this parameter should be set to 0. For RGB->YUV, the recommended value is -0.148 (decimal) or 0x113

14.9.19 G2SB_CTX7_G2CMKEYL_0

When key generation is enabled, the value of this register is the lower color/chroma limit and G2CMKEYU is the upper color/chroma limit for component pixel data. Three component signal values (YCrCb/YUV or RGB) of every output pixel are compared to a set of three ranges that are specified by three pairs of lower and upper color/chroma key values.

If the Key Polarity is 0, the Key is set to 1 only when all three component values of an input pixel are respectively within the set of ranges (inclusive of the limit values), else the Key is set to 0.

If the Key Polarity is 1, the Key is set to 1 when any of the three component values of an input pixel are outside the set of ranges, else the Key is set to 0.

StretchBLT, Color/Chroma Key Lower Limit Register

Offset: 0xa018 | Byte Offset: 0x28060 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRL:R or Cr Color Chroma Key Lower Limit (CKRL[7:0]) Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGL:G or Cb Color Chroma Key Lower Limit (CKGL[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
7:0	X	CKBL: B or Y Color/Chroma Key Lower Limit (CKBL[7:0])

14.9.20 G2SB_CTX7_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0xa019 | Byte Offset: 0x28064 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CKRU: R or Cr Color Chroma Key Upper Limit (CKRU[7:0]) Cr signal must be treated in offset binary format so that the binary interpretation retains monotonicity from the minimum signal level to the maximum signal level.
15:8	X	CKGU: G or Cb Color/Chroma Key Upper Limit (CKGU[7:0]) Cb signal must be treated in offset binary format so that the binary interpretation retains monotonicity the minimum signal level to the maximum signal level.
7:0	X	CKBU:This is B or Y color/chroma key upper limit value.

14.9.21 G2SB_CTX7_G2UBA_A_0

StretchBLT, Source Image Data U base address A-buffer

Offset: 0xa01a | Byte Offset: 0x28068 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SU1SA: Start Address of Source U-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.9.22 G2SB_CTX7_G2VBA_A_0

Offset: 0xa01b | Byte Offset: 0x2806c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SV1SA: Start Address of Source V-image Area, 4:2:0 Format. This parameter specifies the start address of source image stored in the image buffer memory. The [3:0] bits have to be 0, since memory client will assemble YUV into one 4:2:2 format.

14.9.23 G2SB_CTX7_G2SBFORMAT_0

Offset: 0xa01c | Byte Offset: 0x28070 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12:8	X	DIFMT:Destination Image Data Format This parameter defines the data format of destination output. There are two groups of data formats, RGB and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement. 00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 01000: bpp16 5-6-5 {R[4:0], G[5:0], B[4:0]} 01001: RESERVED

Bit	Reset	Description
		01010: RESERVED 01011: RESERVED 01100: bpp16 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]} 01101: RESERVED 01110: R8G8B8A8 01111: B8G8R8A8 1xxxx: RESERVED 0 = U8Y8V8Y8_OB 1 = Y8U8Y8V8_OB 2 = Y8V8Y8U8_OB 3 = V8Y8U8Y8_OB 4 = U8Y8V8Y8_TC 5 = Y8U8Y8V8_TC 6 = Y8V8Y8U8_TC 7 = V8Y8U8Y8_TC 8 = B5G6R5 9 = RESERVED9 10 = RESERVED10 11 = RESERVED11 12 = B5G6R5BS 13 = RESERVED13 14 = R8G8B8A8 15 = B8G8R8A8 16 = RESERVED16 17 = RESERVED17 18 = RESERVED18 19 = RESERVED19 20 = RESERVED20 21 = RESERVED21 22 = RESERVED22 23 = RESERVED23 24 = RESERVED24 25 = RESERVED25 26 = RESERVED26 27 = RESERVED27 28 = RESERVED28 29 = RESERVED29 30 = RESERVED30 31 = RESERVED31
4:0	X	SIFMT: This parameter defines the data format of source input. There are two groups of data formats, RGB format and YCbCr (YUV) format. CbCr (UV) components may be represented in either offset binary or 2's complement. 00000: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00001: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00010: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00011: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V in offset binary format 00100: U8Y8V8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00101: Y8U8Y8V8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00110: Y8V8Y8U8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 00111: V8Y8U8Y8, YUV 4:2:2, 8-bit for each component, U/V is 2's complement 01000: B5G6R5 5-6-5 {R[4:0], G[5:0], B[4:0]} 01001: RESERVED 01010: RESERVED 01011: RESERVED 01100: B5G6R5 5-6-5 Byte-swapped {G[2:0], B[4:0], R[4:0], G[5:3]} 01101: RESERVED 01110: R8G8B8A8 01111: B8G8R8A8 1xxxx: RESERVED **StretchBlit Inputs:** RGB inputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV420

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>input = {YUV420 is converted into 4:2:2 UYVY via memory client} YUV422 inputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}

StretchBlit Outputs: RGB outputs = {B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8} YUV422 outputs = {U8Y8V8Y8_OB, Y8U8Y8V8_OB, V8Y8U8Y8_OB, U8Y8V8Y8_OB, U8Y8V8Y8_TC, Y8U8Y8V8_TC, Y8V8Y8U8_TC, V8Y8U8Y8_TC}

StretchBlit Input/Output Rules
<table>
<tr><th>src format</th><th>internal sb format</th><th>dst format</th></tr>
<tr><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td><td>R8G8B8A8</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*</td></tr>
<tr><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8</td></tr>
</table>
StretchBlit color space converter supports YUV->RGB, YUV->YUV (gain) and RGB->RGB (gain). There is no support for RGB->YUV.
0 = U8Y8V8Y8_OB
1 = Y8U8Y8V8_OB
2 = Y8V8Y8U8_OB
3 = V8Y8U8Y8_OB
4 = U8Y8V8Y8_TC
5 = Y8U8Y8V8_TC
6 = Y8V8Y8U8_TC
7 = V8Y8U8Y8_TC
8 = B5G6R5
9 = RESERVED9
10 = RESERVED10
11 = RESERVED11
12 = B5G6R5BS
13 = RESERVED13
14 = R8G8B8A8
15 = B8G8R8A8
16 = RESERVED16
17 = RESERVED17
18 = RESERVED18
19 = RESERVED19
20 = RESERVED20
21 = RESERVED21
22 = RESERVED22
23 = RESERVED23
24 = RESERVED24
25 = RESERVED25
26 = RESERVED26
27 = RESERVED27
28 = RESERVED28
29 = RESERVED29
30 = RESERVED30
31 = RESERVED31</td></tr>
</table>

14.9.24 G2SB_CTX7_G2CONTROLSB_0

Offset: 0xa01d │ Byte Offset: 0x28074 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000xxx)

Bit	Reset	Description
31	0x0	DISDW: Output destination writes go either to image memory or the EPP. 0: Output data is sent to memory 1: YUV or RGB data is passed directly to the EPP module, and no destination writes take place. 0 = GOTO_IMAGE_BUFFER 1 = GOTO_EPP
30	0x0	ENDITH: Enable Dithering (ENDITH). For 16-bit RGB output modes, the LSB of the color components can be modified by adding a variable residual value that will reduce the banding artifacts that can appear on the display. 0: Normal operation 1: Enable Dithering 0 = DISABLE 1 = ENABLE
28	0x0	KPOL: Key Signal Polarity (KPOL) Color/Chroma key signal is generated by comparing source input pixel color to a range of a color specified by lower and upper limit values. The key signal is interpreted in two ways, depending on which one of video and graphics images is foreground (and the other is background). This is effective only if Key signal generator is enabled. (see G2CMKEYL, G2CMKEYU) 0: Key signal is set to 1 when source pixel is within the lower and upper limit color range. 1: Key signal is set to 1 when source pixel is outside the lower and upper limit color range. 0 = WITHIN_BOUNDS 1 = OUTSIDE_BOUNDS
27	0x0	KEYEN: Key Signal Generator Enable (KEYEN). Key signal generator generates either chroma key signal (from YCbCr signal) or color key signal (from RGB signal) 0: Key signal generator is disabled. 1: Key signal generator is enabled. 0 = DISABLE 1 = ENABLE
26	0x0	DBSEL: StretchBLT Destination Buffer Selection (DBSEL). StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the destination video area. This bit selects one of the two buffering blocks to which this StretchBLT command delivers the destination image. The two buffering memory blocks are called A and B. 0: Destination image goes to ‘A’ buffer. 1: Destination image goes to ‘B’ buffer. 0 = DST_A 1 = DST_B
25	0x0	SBSEL: StretchBLT Source Buffer Selection (SBSEL) StretchBLT processing involves frame-rate conversion from a series of source images to another series of destination images (field-rate of the source video to frame-rate of the PC display). In order to avoid image tearing, it is preferred to use two buffer sections in the source video area. This bit selects one of the two buffering blocks from which this StretchBLT command receives the source image. The two buffering memory blocks are called source-A and source-B. 0: Source image comes from ‘source-A’ buffer. 1: Source image comes from ‘source-B’ buffer. 0 = SRC_A 1 = SRC_B

Bit	Reset	Description
24	0x0	SITYPE: StretchBLT Source Type (SITYPE). This bit identifies two types of source images. If source image is 2-to-1 interlaced and StretchBLT processes either one of the two interlaced field-images, physical (positional) displacement between the two interlaced fields must be taken into account. One field-image that is placed higher in position than the other field-image is called top-field and the other is called bottom-field. StretchBLT processing has to lower the top-field (or raise the bottom-field) to match the corresponding two target images in the overlay window (that is progressively scanned) right at the same position. If source image is full frame-image obtained from two interlaced field-images, its type is 'top-field'. If source images are progressively scanned, the type identification is not significant and they may be designated either one of the two types 0: Source image is 'top-field'. 1: Source image is 'bottom-field'. 0 = TOP_FIELD 1 = BOTTOM_FIELD
23	0x0	RANGEREDFRM:In the VC-1 specification, when the value of RANGEREDFRM variable (1-bit) for a picture is equal to 1, the picture shall be scaled up according to the following equation:Y = clip(((Y-128)*2) + 128);Cb = clip(((Cb-128)*2) + 128);Cr = clip(((Cr-128)*2) + 128);The clip operator limits the output to [0, 255].The input is also limited to [0, 255]. In the VC-1 nomenclature, the output (Y, Cb, Cr) tuple corresponds to the 'decoded' picture. The input (Y, Cb, Cr) tuple corresponds to the 'reconstructed' picture. The above equations create the decoded picture while keeping the reconstructed picture intact.Only YUV _OB formats are supported with range reducation enabled, not the YUV _TC or RGB formats. The YUV data coming from the video frame buffers (reconstructed picture) in the SDRAM was scaled up using the above equations and given to the display. the value of 1-bit wide RANGEREDFRM variable was configured in the DVO per frame and the frame was scaled whenever RANGEREDFRM == 1. 0 = DISABLE 1 = ENABLE
22:20	0x0	HFTYPE: StretchBLT Horizontal Filter Mode (HFTYPE[2:0]) The six-tap horizontal interpolation filter can be operated in various operation modes. For the image expansion, it should be programmed as a pure 6-tap interpolator. For the image contraction, it can work as partly lowpass filter and partly interpolater with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of horizontal filtering. 000: Pure interpolation filter. 001, 010, 011, 100, 101, 110: mix of interpolation and low pass filters 111: DISABLE. 0 = INTERP 1 = LPF1 2 = LPF2 3 = LPF3 4 = LPF4 5 = LPF5 6 = LPF6 7 = DISABLE
19	0x0	DISCSC: enabled color space converter by default 0 = ENABLE 1 = DISABLE
18	0x0	VFEN: StretchBLT Vertical Filter Enable (VFEN). Vertical filter shall be disabled if source images come from host CPU via CPU Read-FIFO. (SISEL) Vertical filter may be disabled to save some memory bandwidth but this will likely result in degradation of image quality. This option may also be used in the case where source image is progressive scanning and there is no vertical scaling. 0: Vertical filter is disabled. 1: Vertical filter is enabled. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
17:16	0x0	VFTYPE: StretchBLT Vertical Filter Mode (VFTYPE[1:0]). This bit is effective when the Vertical Filter is enabled. (bit 18) The two-tap vertical interpolation filter can be operated in various modes. For the image expansion, it should be programmed as a pure 2-tap interpolator. For the image contraction, it can work as partly averager and partly interpolator with varying degree depending on the contraction ratio. For StretchBLT, this parameter is used as an index (selection) to an internal lookup table that stores the group of filter-coefficients for the different modes of vertical filtering. 00: Pure interpolation filter. 01: 25% averager, 75% interpolator. 10: 50% averager, 50% interpolator. 11: 100% averager. 0 = INTERP 1 = AVG25_INTERP75 2 = AVG50_INTERP50 3 = AVG
9:8	X	UVST: 00= 1/2 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 2 bytes. 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case Luma Buffer Stride should be a multiple of 4 bytes. 11= use uvstride. Note: Use care when using a tiled surface, since 1/2 or 1/4 of luma stride may yield an invalid pitch (stride). 0 = UVS2X 1 = UVS1X 2 = UVS4X 3 = UVS_G2UVSTRIDE
7	0x0	ENAHF: enable horizontal alpha filtering if disabled, use the alpha value of third tap for output pixel U,V line stride in 4:2:0 Format. 0 = DISABLE 1 = ENABLE
6	0x0	ENAVF:enable alpha vertical filter 0 = DISABLE 1 = ENABLE
5	0x0	IMODE: Source (input) Data Mode (IMODE). This parameter defines the data mode of source input. 0: Multiplexed Mode (data format is specified by SIFMT[2:0]). One block of source data in the Image Buffer memory contains multiplexed component signals. 1: Planar mode, 4:2:0 YUV (SIFMT[2] specifies data format of U/V components). Three blocks of source data in the Image Buffer memory contain separate Y, U, and V component signals. For a circular buffer input to gr2d, the input format cannot be planar. It must be multiplex, i.e., no YUV420 planar circular buffer input to gr2d. 0 = MULTIPLEX 1 = PLANAR
4	0x0	YUV422PLANAR: Source data is yuv422 planar valid when imode==PLANAR
3	0x0	YUV422ROTATION: YUV422 planar rotated.

14.9.25 G2SB_CTX7_G2CONTROLSECOND_0

Offset: 0xa01e │ Byte Offset: 0x28078 │ Read/Write: R/W │ Reset: 0x00000000 (0bxx000000x00xxxx000xxxx000000000x)

Bit	Reset	Description
29	0x0	FR_READWAIT: Fast Rotate wait for reads. Enabling this bit forces FR to wait for the reads to be stored in the data return FIFO before sending writes out to the same block. Always set to ENABLE. This field is debug only. 0 = DISABLE 1 = ENABLE
28:26	0x0	FR_TYPE: Fast Rotate type 000 = FLIP_X 001 = FLIP_Y 010 = TRANS_LR (mirrors about diagonal. Diagonal runs from upper left to lower right) 011 = TRANS_RL (mirrors about diagonal. Diagonal runs from upper right to lower left) 100 = ROT_90 (counterclockwise by 90 degrees) 101 = ROT_180 110 = ROT_270 (clockwise by 90 degrees) 111 = IDENTITY 0 = FLIP_X 1 = FLIP_Y 2 = TRANS_LR 3 = TRANS_RL 4 = ROT_90 5 = ROT_180 6 = ROT_270 7 = IDENTITY
25:24	0x0	FR_MODE: Fast Rotate mode sel G2 Fast Rotate Transforms a surface via FR_TYPE transformation. Works in either 2-buffer (copy) or 1-buffer mode (in place) mode. The engine breaks down a larger surface into a grid of smaller FR_BLOCKs. Works on the granularity of an FR_BLOCK. An FR_BLOCK is: ▪ 16x16 pixel block (DSTCD = bpp8) ▪ 8x8 pixel block (DSTCD = bpp16) ▪ 4x4 pixel block (DSTCD = bpp32) Maximum surface size is 4096x4096 Key information - source and destination base address must be 128-bit word aligned - engine works on FR_BLOCK granularity: transformed surface width in multiples of 16 bytes** transformed surface height in multiples of 16/8/4 lines for bpp8/bpp16/bpp32 FR_BLOCK if surface dimension is not a multiple, software can program FR engine to transform larger surface (round up to next FR_BLOCK in width and height-->transform-->calculate relative memory pointer address) - during a rotational transformation (TRANS_LR, TRANS_RL, ROT_90, ROT_270): the stride of the output surface != the stride of the input surface when working on non-square input input: output: *^---------- *$---------- $ 0 1 2 3 4 5 6 7 ^ 24 16 8 0 ^ ^ ^ ^ | 8 9 10 11 12 13 14 15 | 25 17 9 1 ^ ^ ^ ^ |16 17 18 19 20 21 22 23 | 26 18 10 2 ^ ^ ^ ^ |24 25 26 27 28 29 30 31 | 27 19 11 3 ^ ^ ^ ^ | - - - - * * * * | 28 20 12 4 * * * * | - - - - * * * * | 29 21 13 5 * * * * | - - - - * * * * | 30 22 14 6 * * * * | - - - - * * * * | 31 23 15 7 * * * * - Tiling alignment restrictions subsume FR restrictions because the FR surface base address is the same as the start address. **Register Programming** FR_MODE - inplace or copy FR_TYPE - type of transformation DSTCD - bpp8, bpp16, bpp32 SRCBA - source base address SRCWIDTH - (width in pixels-1) SRCHEIGHT - (height in lines-1) SRCS - source stride DSTBA - destination base address

Bit	Reset	Description
		DSTS - destination stride FR_READWAIT - always set to enable FR inefficency in the following setup: 1. FR_MODE==SQUARE 2. if(FR_TYPE==YFLIP && SRCHEIGHT==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 Or if(FR_TYPE==XFLIP && SRCWIDTH==m*n && n==odd number) where m=8bpp?16:16bpp?8:4 3. Then, the middle 8 lines/pixels, e.g., YFLIP/16bpp/n==3, line 8 to line 15 will be processed twice. 00 = disable fast rotate - this turns off the second level clock to fr_rotate engine. Please remember to do so! 01 = src/dst copy mode - two separate buffers 10 = square in place - one buffer 11 = blank 0 = DISABLE 1 = SRC_DST_COPY 2 = SQUARE 3 = BLANK
22:21	0x0	CLIPC: Clipping rectangle control, if clip enable, bit 57, XYTDW should be cleared. 0x=clipping disabled 10=draw only inside clipping rectangle 11=draw only outside clipping rectangle
9	0x0	ALPSRCORDST: 32-bit blending mode, output alpha selection 0:source alpha 1:destination alpha 0 = DISABLE 1 = ENABLE
8:4	0x0	ALPTYPE: Alpha blending method FIX: ALPHA blending: Fixed alpha, ALPHA is the value, B5G6R5 * B5G6R5 VCAA: B5G6R5->B5G6R5 PL1BPP: ALPHA blending: Alpha 1 bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero. PL2BPP: ALPHA blending: Alpha 2 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero. PL4BPP: ALPHA blending: Alpha 4 bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero. PL8BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8 bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1 bit from source B5G5R5A1, alpha(MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4 bits from source A4B4G4R4, alpha(LSB). Dest: B5G6R5. PLS4BPP: ALPHA blending: Alpha 4 bits from source B4G4R4A4, alpha(MSB). Dest: B5G6R5. VCAA: R8G8B8A8->B5G6R5 without reading VCAA plane (surface blit with bpp down convert - implemented in hardware by the VCAA engine; not really a VCAA resolve) PLS8BPP: ALPHA blending: Alpha 8 bits from source/destination (decided by ALPSRCORDST), R8G8B8A8, alpha(MSB). DST: R8G8B8A8 VCAA: R8G8B8A8->R8G8B8A8, alpha has same blending method as RGB PLS8BX: ALPHA blending: Alpha 8 bits from source B8G8R8A8, alpha (MSB). Dest: B5G6R5. (**Restrictions) VCAA: R8G8B8A8->B5G6R5 PLS1BPPAL: ALPHA blending: Alpha 1 bit from source A1B5G5R5, alpha (LSB). Dest: B5G6R5 VCAA: A1B5G5R5->A1B5G5R5 **Restriction PLS8BX alpha blending has the following restrictions: 1. Source/destination addresses have to be in a 128-bit boundary. 2. Destination width has to be a multiple of 4 pixels. 3. Source/Destination strides have to be a multiple of 128 bits. 0 = FIX

Bit	Reset	Description
		1 = PL1BPP 2 = PL2BPP 3 = PL4BPP 4 = PL8BPP 5 = PL44BPP 6 = PLS1BPP 7 = PLS4BPPAL 8 = PLS4BPP 9 = PLS8BPP 10 = PLS8BX 11 = PLS1BPPAL
3	0x0	BEWSWAP: Host port word swap 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 0 = DISABLE 1 = ENABLE

14.9.26 G2SB_CTX7_G2CONTROLMAIN_0

G2 Blit Formats:

(1) G2 Blit size is based on height in lines, width in pixels, stride in bytes, and pixel size (color depth). Unless alpha blending is enabled, G2 does not care about pixel components.

Source color depths supported by G2 Blit:

- color depth same as destination: G2CONTROLMAIN.SRCCD = 1
- monochrome: G2CONTROLMAIN.SRCCD = 0

(source monochrome color determined by G2SRCBGC, G2SRCFGC)

Pattern color depths supported by G2 Blit:

- color depth same as destination: G2PATOS.PATCD = 1
- monochrome: G2PATOS.PATCD = 0

(pattern monochrome color determined by G2PATBGC, G2PATFGC)

Destination color depths supported by G2 Blit:

- 1 byte per pixel: G2CONTROLMAIN.DSTCD = BPP8
- 2 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP16
- 4 bytes per pixel: G2CONTROLMAIN.DSTCD = BPP32

(2) Alpha blending operates on 3 or 4 component pixels of a specified color depth. One of the components may be alpha (A), depending on the alpha blend format, which is only allowed in certain component positions defined below. The only format rule of the remaining components is that they must be in the same source and destination position because the alpha blend engine does not support component swapping. For example, valid 32BPP formats are: RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888, GRBA_8888->GRBA_8888, etc.

Source formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16:

- ALPTYPE_PLS1BPP: xxxA_5551 (for example, BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
- ALPTYPE_PLS4BPP: xxxA_4444 (for example, BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
- (other) : xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])

- color depth BPP32:
  - (all) : xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

Destination formats supported by G2 alpha blend:

- color depth BPP8: no supported formats
- color depth BPP16: xxx_565 (for example, BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (for example, BRGA_8888 = A[31:24] R[23:16] G[15:8] B[7:0])

src cd	src format	dst cd	dst format	examples
BPP8	no support	BPP8	no support	
BPP16	xyzA_5551, xyzA_4444, xyz_565	BPP16	xyz_565	BGRA_5551->BGR_565, RGB_565->RGB565
BPP32	xyzA_8888	BPP32	xyzA_8888	RGBA_8888->RGBA_8888, BGRA_8888->BGRA_8888
BPP32	xyzA_8888	BPP16	xyz_565	RGBA_8888->RGB_565, BGRA_8888->BGR_565

VCAA Resolve Engine

(1) Format support

srcColor format	srcCoverage format	legal dest format(s)
A1B5G5R5	VCAA	A1B5G5R5
R8G8B8A8	VCAA	R8G8B8A8, B5G6R5
B5G6R5	VCAA	B5G6R5

To program resolve type:

- ALPTYPE == FIX: B5G6R5 -> B5G6R5
- ALPTYPE == PLS1BPPAL: A1B5G5R5 -> A1B5G5R5
- ALPTYPE == PLS8BPP: R8G8B8A8 -> R8G8B8A8
- ALPTYPE == PLS8BX: R8G8B8A8 -> B5G6R5

(2) Color surface programming

- color surface base address: SRCBA
- sub-surface start x: SRCX (pixel index)
- sub-surface start y: SRCY (line index)
- color surface stride: SRCS (bytes)

The engine supports sub-surface resolve. One can imagine a full color buffer of 1024x768, but

```
[pseudo_code]
      color_surface_start_address_fetch = pcurctx->regs.rG2SRCBA.uSRCBA() +
        (pcurctx->regs.rG2SRCPS.uSRCX() * VCAAState.color_surface_depth) +
        (pcurctx->regs.rG2SRCPS.uSRCY() * VCAAState.color_surface_stride);
```

(3) VCAA surface programming

- VCAA surface base address: PATBA
- VCAA surface stride: PATST (bytes)

```
[pseudo_code]
      vcaa_surface_start_address_fetch = pcurctx->regs.rG2PATBA.uPATBA();
```

Software *MUST* directly program this register to the proper sub-surface location in the vcaa surface which corresponds to the SRCX and SRCY programming. The calculation is PATBA = vcaa_base_address + SRCX * (1) + SRCY * PATST;

(4) Resolve surface programming

- resolve window width: DSTWIDTH (pixels)
- resolve window height: DSTHEIGHT (lines)
- resolve output stride: DSTS (bytes)

These values program the width and height of the resolved surface. In the base case,

- DSTWIDTH = color surface width
- DSTHEIGHT = color sufrace height
- DSTS = resolve surface format bpp * DSTWIDTH

The VCAA engine technically supports DSTX and DSTY (like SRCX and SRCY).

```
[pseudo_code]
   resolve_surface_start_address_put = pcurctx->regs.rG2DSTBA.uDSTBA() +
     (pcurctx->regs.rG2DSTPS.uDSTX() * resolve_surface_color_depth) +
     (pcurctx->regs.rG2DSTPS.uDSTY() * resolve_surface_color_stride);
```

(5) Resolving a pixel

The coverage surface is an 8 bits per pixel surface (C4X4). The surface is initialized to all 1s by 3d.

For a given pixel, we need to look up 4 bits of coverage data. If the data is 0xf, the pixel does not need to be resolved. Otherwise, the pixel needs to be reblended with certain weightings of its neighbors (i.e., we are on the edge of some geometry).

If any coverage bits are 0, the color needs to be resolved. To do so, calculate the following equation:

```
color_new = (
              20*color_old +
              27*(
                  ((cover_down  == 0) ? color_down  : color_old) +
                  ((cover_right == 0) ? color_right : color_old) +
                  ((cover_left  == 0) ? color_left  : color_old) +
                  ((cover_up    == 0) ? color_up    : color_old)
                  )
            ) / 128;
```

If coverage bit == 0, then use the neighbor color value in the resolve.

If coverage bit == 1, then use the center color value in the resolve.

The resolve is done on a per channel (r/g/b/a) basis:

format convert input channel to 8-bit

perform resolve equation above

format convert to output format

For the resolution of the alpha channel:

- In B5G6R5: No alpha information. Do nothing.
- In A1B5G5R5: No resolve. Keep the source color alpha bit.

- In R8G8B8A8: Normal 8-bit channel resolve

(6) Resolving a pixel on the edge

When on an edge, the VCAA engine treats the coverage bit as a 1. Use the center color value. This holds true even on the edge of the sub-surface resolve window when located in the middle of the larger surface

(7) Maximum dimensions

The maximum resolve subsurface is 4096 pixels wide.

(8) Surface restrictions

The VCAA engine has the following restrictions with respect to input surfaces

- input base address is 128-bit memory word aligned

- input stride is a multiple of 128 bits (16 bytes)

The VCAA engine has the following restrictions with respect to output surfaces

- output base address is 128-bit memory word aligned

- output stride is a multiple of 128 bits (16 bytes)

(9) Coverage surface layout

The surface is C4X4.

Surface layout (C4X4)

```
(lsb2msb)  C0_C1_C2_C3_X_X_X_X     (X is a don't care.  Qrast inits to 1)
(msb2lsb)  X_X_X_X_C3_C2_C1_C0

    +          o          o
         C0
            C1   Cov(0,1)
       C2
         C3
    o          o          o

      Cov(1,0)   Cov(1,1)

    o          o          o

    +/o = pixel centers
```

In the example above,

```
Cov(0,0) = X_X_X_X_C3_C2_C1_C0
C3       = V_C3(Cov(0,0));
..         ..
C0       = V_C0(Cov(0,0));
```

```
#define V_C3(bits) (bits & 0x8)
#define V_C2(bits) (bits & 0x4)
#define V_C1(bits) (bits & 0x2)
#define V_C0(bits) (bits & 0x1)
```

Resolving a Pixel

In an ideal world, the coverage would be simply laid out such that when resolving Pixel(x,y), we would need only to fetch Cov(x,y) to get all 4 bits of information.

In actuality, the coverage surface was rearranged to help qrast in performance. To resolve Pixel(x,y) gr2d has to fetch 4 C4X4 bytes [Cov(x,y), Cov(x-1,y), Cov(x,y-1), Cov(x-1,y-1)]. From each C4X4 byte, it extracts one bit which is used in the resolve equations shown in (5).

Need the logical VCAA bits (DRLU) for blend which is spread across four C4X4 coverage bytes

```
      U
  L
    o
      R
    D
```

where

o = Pixel(x,y)

D = V_C1(Cov(x-1,y))

R = V_C0(Cov(x,y))

L = V_C3(Cov(x-1,y-1))

U = V_C2(Cov(x,y-1))

Offset: 0xa01f | Byte Offset: 0x2807c | Read/Write: R/W | Reset: 0x00000000 (0bx000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: Pattern Data Select: SRCSEL and PATSEL cannot be both enabled. 0 = SCREEN 1 = MEMORY
28	0x0	DSTDIR:destination direct addressing
27	0x0	SRCDIR:source direct addressing
26:25	0x0	GCSW: Display Switching Window Control (GCSW[1:0]). This parameter controls multi-buffering for Display. x0: At end of current command, do not send signal to Display to switch buffer. 01: two buffers, dstba and dstba_b are used 11: three buffers, dstba, dstba_b, dstba_c are used
24	X	SRCBAS: Source base address select: 0=srcba 1=dstba. This is not used for Line Draw and if source data comes from host memory.
23:22	0x0	SRCT: Source transparency enable: 0x=source transparency disabled, 10=mono source background transparency or color source transparency,

Bit	Reset	Description
		11=mono source foreground transparency or inverse color source transparency. NOTE: When source transparency is enabled and SRCCD==0(mono) SRCBGC!=SRCFGC should be satisfied.
21	X	HLMONO: Start from Msb or lsb in byte when mono expansion If HLMONO is 1, bit 0 (the lsb) is the first bit. If HLMONO is 0, bit 7 (the msb) is the first bit
20	X	SRCCD: 0: Source mono 1: Source has same color depth as destination SRCCD==0 xdir/ydir has to be 0
19:18	0x0	DSTT: Destination read transparency enable: 0x=destination read transparency disabled 10=color destination read transparency 11=inverse color destination read transparency. 20 rw INIC Initiate Command (1=initiate command immediately, 0=wait for launch write)
17:16	X	DSTCD: Destination color depth: 00=8-bpp, 01=16-bpp, 10=32-bpp. 11=reserved 0 = BPP8 1 = BPP16 2 = BPP32 3 = RESERVED3
15	0x0	SRCSEL: Source Data Select: 0 = SCREEN 1 = MEMORY
14	0x0	YFLIP: flip y direction to make image upside down or the other way. If YFLIP==1, ROP cannot include destination. 0 = DSIABLE 1 = ENABLE
13	0x0	PATPACK: Pattern Data is in Pack Mode. PATLNGAP in G2PATPACK is the line gap for pattern packed mode If (PATPACK && ~PATSEl), pattern data is packed and from screen, PATMONOW/PATMONOH should be programmed properly to fetch pattern data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
12	0x0	SRCPACK: Source Data is in Pack Mode. SRCLNGAP in G2SRCPACK is the line gap for source packed mode. If (SRCPACK && ~SRCSEL), source data is packed and from screen, SRCMONOW/SRCMONOH should be programmed properly to fetch data from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE
11	0x0	XYTDW: xy transpose, Line stride DSTS has to be 16-byte aligned if enabled. If XYTDW==1, ROP cannot include destination.
10	0x0	YDIR: 0=incrementing, 1=decrementing. ydir should not be set when source surface has different color depth as destination surface. SRCCD==0 (mono src) OR PATCD==0 (mono pat), ydir has to be 0
9	0x0	XDIR: 0=incrementing, 1=decrementing. xdir should not be set when source surface has different color depth as destination surface. e.g. ALPTYPE=PLS8BX (32bpp blending with 16bpp), xdir has to be 0 SRCCD==0 (mono src) OR PATCD==0 (mono pat), xdir has to be 0

Bit	Reset	Description

8	0x0	PATFL: When mono pattern is set, we use mono tile pattern fill. current pattern client can support a 16x16 bit tile which can be used to generate a larger surface (i.e. tile this surface in the x and y direction) The tile is stored in memory. Currently, PATXO and PATYO indicate where to start in the 16x16 tile when expanding the monochrome data. See patxo and patyo comments below for programming
		When xdir==1 Patxo = pattern width - (patxo+destination width)&0xF -- For xoffsets y offsets PATYO remains the same when xdir/ydir change. How the tile replication pattern looks based on xdir/ydir (without programming patxo above): xdir=0 ydir=0 xdir=1 ydir=0 +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ +---+--+ +--+---+ |^|^|^|| ||^|^|^| |~~~|~~| |~~|~~~| +---+--| |--+---+ |^|^|^|| ||^|^|^| +------+ +------+ // xdir=0 ydir=1 xdir=1 ydir=1 Mono tile is +---+ |^|^| |~~~| +---+
		0 = DISABLE
		1 = ENABLE

7	0x0	PATSLD: BitBlt Solid Pattern Fill: 1=enable. BGC will be used as the color value. 0 = DISABLE 1 = ENABLE
6	0x0	SRCSLD: BitBlt Solid Source Color Fill: 1=enable. FGC will be used as the color value. 0 = DISABLE 1 = ENABLE
5	0x0	ALPEN: BitBlt Alpha Blending, 1=enable. 0=disable,when both Faden and alpen are 1, output=Source*alpha_v + fadoff, alpha_v is decided by alptype
4	0x0	FADEN: BitBlt Source Copy Fade enable, 1=enable (share with mltln), only support 16bpp mode 0 = DISABLE 1 = ENABLE
3	0x0	TEST0BIT: Command finish timing bit 0: 2D command finishes when last data has been pushed to memory write client. 1: 2D command waits memory write client to be idle to finish. 0 = DISABLE 1 = ENABLE
2	0x0	TURBOFILL: fast fill rectangle in 128 bits/clock. Some limitations with this mode:srcsld==1 rop==0xcc, no clipping, no transparency, xdir==0, ydir==0, flip==0, xytdw==0. Results are undefined if the above limitations are not satisfied.
1:0	0x0	CMDT: Command Type: 00=BitBlt 01=Line Draw 10=VCAA 11=reserved When the raise command is executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

14.9.27 G2SB_CTX7_G2ROPFADE_0

Offset: 0xa020 │ Byte Offset: 0x28080 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31:24	X	FADOFF:
23:16	X	FADCOE:
7:0	0x0	ROP: If YFLIP==1 or XYTDW==1, ROP cannot include destination. Since destination may have been corrupted before reading out.

14.9.28 G2SB_CTX7_G2ALPHABLEND_0

Offset: 0xa021 | Byte Offset: 0x28084 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
31:24	X	ALPHA1V
23:16	X	ALPHA0V
8	0x0	ALPHAINV
7:0	X	ALPHA

14.9.29 G2SB_CTX7_G2CLIPLEFTTOP_0

ClipRect Methods. Top-left bounds are inclusive

Offset: 0xa022 | Byte Offset: 0x28088 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPT
14:0	X	CLIPL

14.9.30 G2SB_CTX7_G2CLIPRIGHTBOT_0

Bottom-right bounds are exclusive

Offset: 0xa023 | Byte Offset: 0x2808c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	CLIPB
14:0	X	CLIPR

14.9.31 G2SB_CTX7_G2PATPACK_0

Pattern packed mode

Pattern methods. G2PATPACK should be used to specify the line gap. Use G2PATPACK_SIZE to program height and width. PACK is generally only useful with narrow monochrome surfaces

Note: PACK is not officially supported when the surface is tiled.

Offset: 0xa024 | Byte Offset: 0x28090 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

14.9.32 G2SB_CTX7_G2PATPACK_SIZE_0

Pattern packed mode

Packed mode, pattern data line gap. Byte

G2PATPACK_SIZE -- extension of G2PATPACK, this register only holds the size of the packed data

Note: PACK is not officially supported when the surface is tiled.

Offset: 0xa025 | Byte Offset: 0x28094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	PATMONOH: Pattern mono data height in packed mode, byte
15:0	X	PATMONOW: Mono data width in packed mode, byte

14.9.33 G2SB_CTX7_G2PATBA_0

Offset: 0xa026 | Byte Offset: 0x28098 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBA: If (PATFL==1){ It has to be 16-byte aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

14.9.34 G2SB_CTX7_G2PATOS_0

Offset: 0xa027 | Byte Offset: 0x2809c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00xxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	PATYO: y offset for mono tile pattern fill. see PATFL
27:24	X	PATXO: x offset for mono tile pattern fill. see PATFL
22:21	0x0	PATT: Mono pattern transparency enable: 0x=pattern transparency disabled, 10=mono pattern background transparency or color pattern transparency, 11=mono pattern foreground transparency or inverse color pattern transparency. NOTE: When pattern transparency is enabled and PATCD==0(mono) PATBGC!=PATFGC should be satisfied.
16	X	PATCD:0 mono1 same as dstcd PATCD==0 xdir/ydir has to be 0
15:0	X	PATST:stride

14.9.35 G2SB_CTX7_G2PATBGC_0

Offset: 0xa028 | Byte Offset: 0x280a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

14.9.36 G2SB_CTX7_G2PATFGC_0

Offset: 0xa029 | Byte Offset: 0x280a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

14.9.37 G2SB_CTX7_G2PATKEY_0

Offset: 0xa02a | Byte Offset: 0x280a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

14.9.38 G2SB_CTX7_G2DSTBA_0

Offset: 0xa02b | Byte Offset: 0x280ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA: Destination base address (byte address)

14.9.39 G2SB_CTX7_G2DSTBA_B_0

Offset: 0xa02c | Byte Offset: 0x280b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_B: Destination base address (byte address) only usable in hardware trigger mode by enable gcsw

14.9.40 G2SB_CTX7_G2DSTBA_C_0

Offset: 0xa02d | Byte Offset: 0x280b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSTBA_C: Destination base address (byte address)

14.9.41 G2SB_CTX7_G2DSTST_0

Offset: 0xa02e | Byte Offset: 0x280b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

14.9.42 G2SB_CTX7_G2SRCPACK_0

source data packed mode

Surface Methods

G2SRCPACK should only be used to specify the line gap. Use G2SRCPACK_SIZE to program height and width

Offset: 0xa02f │ Byte Offset: 0x280bc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

14.9.43 G2SB_CTX7_G2SRCPACK_SIZE_0

Source Data Packed Mode

In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, the required width/stride is 16 byte aligned. Packed mode - source mono data line gap

G2SRCPACKS_SIZE -- extension of G2SRCPACK, this register only holds the size of the packed data

Offset: 0xa030 │ Byte Offset: 0x280c0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCMONOH: Packed mode - source mono data height in bytes
15:0	X	SRCMONOW: Packed mode - source mono data width in bytes

14.9.44 G2SB_CTX7_G2SRCBA_0

src base address

Offset: 0xa031 │ Byte Offset: 0x280c4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA: Source base address (byte address)

14.9.45 G2SB_CTX7_G2SRCBA_B_0

SB only

Offset: 0xa032 │ Byte Offset: 0x280c8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBA_B: This parameter specifies the start address of source image stored in the image buffer memory. In 4:2:0 format mode, this image block accommodates for Y-image.This address specifies byte position; however, bits [2:0] are restricted with respect to the data formats to fit multiple pixels in one memory word (8 bytes). For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format. Since one Y pixel takes 8 bits, all 8 byte positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there are no restrictions on this value.)

14.9.46 G2SB_CTX7_G2SRCST_0

Offset: 0xa033 | Byte Offset: 0x280cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS: Source Stride coordinate (bytes) with respect to SRCBA. In order to fit multiple pixels in one memory word (8 bytes), bits [2:0] are restricted with respect to the data formats. For example, {0, 4} for any YCrCb formats, {0, 2, 4, 6} for RGB 16-bit format.

14.9.47 G2SB_CTX7_G2SRCBGC_0

srcColors

Offset: 0xa034 | Byte Offset: 0x280d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

14.9.48 G2SB_CTX7_G2SRCFGC_0

Offset: 0xa035 | Byte Offset: 0x280d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

14.9.49 G2SB_CTX7_G2SRCKEY_0

Offset: 0xa036 | Byte Offset: 0x280d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

14.9.50 G2SB_CTX7_G2SRCSIZE_0

Offset: 0xa037 | Byte Offset: 0x280dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	SRCHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	SRCWIDTH

14.9.51 G2SB_CTX7_G2DSTSIZE_0

Offset: 0xa038 | Byte Offset: 0x280e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:16	X	DSTHEIGHT: In SB mode, number of lines - 1 In 2D mode, actual lines
14:0	X	DSTWIDTH: In 2D mode, the largest number can be programmed is 0x7fef

14.9.52 G2SB_CTX7_G2SRCPS_0

ImageBlit Methods

Offset: 0xa039 | Byte Offset: 0x280e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	SRCY
15:0	X	SRCX:SRCX[2:0] are ignored in SRCCD==0 (mono expansion). The first bit of the first byte (bit 7 if HLMONO==0, or bit 0 if HLMONO==1) always expand to DSTX,DSTY

14.9.53 G2SB_CTX7_G2DSTPS_0

Offset: 0xa03a | Byte Offset: 0x280e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	DSTY
15:0	X	DSTX: NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero

14.9.54 G2SB_CTX7_G2CBDES_0

Circular Buffer

Offset: 0xa03b | Byte Offset: 0x280ec | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TOPCLIP:top clipping at the first buffer, (buffer start) refer to TOP_CBLINE in G2CBDES2 0 = DISABLE 1 = ENABLE
30:16	X	CBLINE:vertical line number in one buffer
7:0	X	CBCOUNT: This specifies the number of buffers incircular buffer feature

14.9.55 G2SB_CTX7_G2CBSTRIDE_0

Offset: 0xa03c | Byte Offset: 0x280f0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	X	CBUVSTRIDE: Chroma Buffer Stride default is half of luma 00= 1/2 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 2 bytes. 01= Equal to Luma Buffer Stride 10= 1/4 of Luma Buffer Stride; in this case, Luma Buffer Stride should be a multiple of 4 bytes. 1x= Reserved 0 = CBS2X 1 = CBS1X 2 = CBS4X
23:0	X	CBSTRIDE: Video Buffer Luma (or RGB) Buffer Stride. This is luma buffer stride (in bytes)

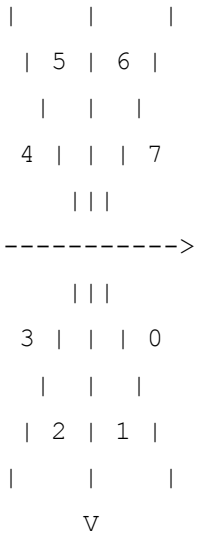
14.9.56 G2SB_CTX7_G2LINESETTING_0

Line Methods

Offset: 0xa03d | Byte Offset: 0x280f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	OCTANTS: 000: octant 0 001: octant 1 010: octant 2 011: octant 3 100: octant 4 101: octant 5 110: octant 6 111: octant 7
28	X	LINEUSEOCTANT:use OCTANTS in G2LINEDELTAN register instead of MAJOR LINEXDIR LINEYDIR
27	X	DROPLASTP: Draw last pixel or not
26	X	LINEYDIR
25	X	LINEXDIR
24	X	MAJOR: 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

14.9.57 G2SB_CTX7_G2LINEDELTAN_0



Offset: 0xa03e | Byte Offset: 0x280f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

14.9.58 G2SB_CTX7_G2LINEDELTAM_0

Offset: 0xa03f | Byte Offset: 0x280fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

14.9.59 G2SB_CTX7_G2LINEPOS_0

Offset: 0xa040 | Byte Offset: 0x28100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LINEYPOS
15:0	X	LINEXPOS

14.9.60 G2SB_CTX7_G2LINELEN_0

Offset: 0xa041 | Byte Offset: 0x28104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

14.9.61 G2SB_CTX7_G2CSCFOURTH_0

Offset: 0xa042 | Byte Offset: 0x28108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:16	X	G2V: Multiplier for G for V generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.368 (decimal) or 0x12F. For any other combination, this parameter is ignored.
8:0	X	G2U: Multiplier for G for U generation. This parameter consists of a sign bit and 8-bit magnitude (s1.7). For RGB->YUV, the recommended value is -0.291 (decimal) or 0x125. For any other combination, this parameter is ignored.

14.9.62 G2SB_CTX7_G2SRCST_B_0

Offset: 0xa043 | Byte Offset: 0x2810c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

14.9.63 G2SB_CTX7_G2UVSTRIDE_0

Offset: 0xa044 | Byte Offset: 0x28110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

14.9.64 G2SB_CTX7_G2CBDES2_0

Circular Buffer Controller 2

Offset: 0xa045 | Byte Offset: 0x28114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

14.9.65 G2SB_CTX7_G2TILEMODE_0

Offset: 0xa046 | Byte Offset: 0x28118 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx0xxx0xxx0xxx0xxx0xxx0)

Bit	Reset	Description
20	0x0	DST_WR_TILE_MODE: destination surface 0 = LINEAR 1 = TILED
16	0x0	DST_RD_TILE_MODE: Same as destination write unless DST_RD_WR_SEP (not supported) 0 = LINEAR 1 = TILED
12	0x0	PAT_UV_TILE_MODE: UNUSED 0 = LINEAR 1 = TILED
8	0x0	PAT_Y_TILE_MODE: Y or RGB surface 0 = LINEAR 1 = TILED
4	0x0	SRC_UV_TILE_MODE: UV surface, ignored in RGB mode 0 = LINEAR 1 = TILED
0	0x0	SRC_Y_TILE_MODE: Y or RGB surface 0 = LINEAR 1 = TILED

14.9.66 G2SB_CTX7_G2PATBASE_0

Offset: 0xa047 | Byte Offset: 0x2811c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PAT_BASE:pattern base address in tile mode, PATBA is the linear address where pixel start

14.9.67 G2SB_CTX7_G2SRCBA_SB_SURFBASE_0

SB_SURFBASE Registers

These registers need only be programmed when using SB. They point to the base of the various source and destination surfaces. Technically, they are only needed when tiling is enabled, but there is no harm in always programming them.

Their counterpart registers (_SB_SURFBASE with SB_SURFBASE stripped off) indicate the location of the first pixel to be sourced or written within the surface.

This register exists to mimic the X, Y (SRCX, SRCY) functionality found in the BitBlt engine.

For example, to get at pixel X, Y:

SRCBA = SRCBA_SB_SURFBASE + Y*stride + X*Bpp

Offset: 0xa048 | Byte Offset: 0x28120 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRC_ADDR: Surface address corresponding to G2SRCBA: -base of interleaved sources (RGB, YUV) -base of Y plane. Only used by the StretchBlit Engine

14.9.68 G2SB_CTX7_G2DSTBA_SB_SURFBASE_0

Offset: 0xa049 | Byte Offset: 0x28124 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_ADDR: surface address corresponding to G2DSTBA. Only used by the StretchBlit Engine

14.9.69 G2SB_CTX7_G2DSTBA_B_SB_SURFBASE_0

Offset: 0xa04a | Byte Offset: 0x28128 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DST_B_ADDR: Surface address corresponding to G2DSTBA_B. Only used by the StretchBlit Engine, and G2CONTROLSB.DBSEL() is enabled

14.9.70 G2SB_CTX7_G2VBA_A_SB_SURFBASE_0

Offset: 0xa04b | Byte Offset: 0x2812c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	V_ADDR: Surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of V plane. Only used by the StretchBlit Engine

14.9.71 G2SB_CTX7_G2UBA_A_SB_SURFBASE_0

Offset: 0xa04c | Byte Offset: 0x28130 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	U_ADDR: Surface address corresponding to G2VBA used for YUV 4:2:0 planar, base of U plane. Only used by the StretchBlit Engine

14.10 G2SB Switch Registers

14.10.1 G2SB_SWITCH_G2INTERRUPT_0

Interrupt Status Register

These registers can be only accessed by register read/write not command FIFO interface. When context switch fails, the CPU received an interrupt from G2.

Context switch steps:

1. Clears G2INTERRUPT/CTXSW_INT bit

2. Read out G2NXTCXTSWITCH/NEXT_CLASS NEXT_CHANNEL to determine failing on which class channel
3. Decide which context can be switched to
4. Back up the context contents that will be overwritten
5. Write channel into G2CLASSCHANNEL_REGONLY/CURR_CHANNEL
6. Command FIFO flow resumes.

Offset: 0xf000 | Byte Offset: 0x3c000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	GR2D_IDLE
0	0x0	CTXSW_INT

14.10.2 G2SB_SWITCH_G2INTENABLE_0

Interrupt Enable Register

Offset: 0xf001 | Byte Offset: 0x3c004 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	GR2D_IDLE_ENABLE
0	0x0	CTXSW_INT_ENABLE

14.10.3 G2SB_SWITCH_G2CURRENTCONTEXT_0

Offset: 0xf002 | Byte Offset: 0x3c008 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	CURR_CONTEXT

14.10.4 G2SB_SWITCH_G2NXTCXTSWITCH_0

When context switch fails, software needs read out this register to find out failing on which class and channel.

Offset: 0xf003 | Byte Offset: 0x3c00c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	NEXT_CHANNEL
9:0	X	NEXT_CLASS

14.10.5 G2SB_SWITCH_G2GLOBALCONTROL_0

initial index of destination address, 3 selections 00 DSTBA 01 DSTBA_B 10 DSTBA_C

Offset: 0xf004 | Byte Offset: 0x3c010 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	DST_ADDR_IDX_INI:31:8 rw CLOCKFREEON init = 0

14.10.6 G2SB_SWITCH_G2GLOBALCONTROLB_0

Offset: 0xf005 | Byte Offset: 0x3c014 | Read/Write: R/W | Reset: 0x000000e1 (0bxxxxxxxxxxxxxxxxxxxxxxxx11100001)

Bit	Reset	Description
7:0	0xe1	OCTAN_BIAS

14.10.7 G2SB_SWITCH_G2WORKINGSTAT_0

Offset: 0xf006 | Byte Offset: 0x3c018 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:4	X	WORKING_CTX
3:0	X	WORKING_CHANNEL

14.10.8 G2SB_SWITCH_G2BUFTHRESHOLD_0

Offset: 0xf007 | Byte Offset: 0x3c01c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	BUFFER_COUNT_THRESHOLD

14.10.9 G2SB_SWITCH_CLKEN_OVERIDE_0

Offset: 0xf008 | Byte Offset: 0x3c020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
22	CLK_GATED	SB_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
21	CLK_GATED	G2PR2MC_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
20	CLK_GATED	G2DR2MC_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
19	CLK_GATED	G2SR2MC_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
18	CLK_GATED	VCAA_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
17	CLK_GATED	LINER_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
16	CLK_GATED	LINE_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
15	CLK_GATED	FR_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON

Bit	Reset	Description
14	CLK_GATED	DSTR_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
13	CLK_GATED	PATR_CLKEN_OVR: It forces cbr_g2pr2mc_clk on too 0 = CLK_GATED 1 = CLK_ALWAYS_ON
12	CLK_GATED	SRCR_CLKEN_OVR: It forces cbr_g2sr2mc_clk on too 0 = CLK_GATED 1 = CLK_ALWAYS_ON
11	CLK_GATED	SRC_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
10	CLK_GATED	BBLT_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
9	CLK_GATED	DSTW_CLKEN_OVR: It forces ccw_g2dw2mc_clk on too 0 = CLK_GATED 1 = CLK_ALWAYS_ON
8	CLK_GATED	CON_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
7	CLK_GATED	CONTEXT7_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
6	CLK_GATED	CONTEXT6_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
5	CLK_GATED	CONTEXT5_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
4	CLK_GATED	CONTEXT4_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
3	CLK_GATED	CONTEXT3_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
2	CLK_GATED	CONTEXT2_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
1	CLK_GATED	CONTEXT1_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
0	CLK_GATED	CONTEXT0_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON

14.10.10 G2SB_SWITCH_G2_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register

Note: The FIFO timing aspects of this register are no longer supported, but retained for software compatibility.

The clock enable fields of this register control the second-level clock gating for the MC side of the MCCIF.

A '1' written to the override field will result in that clock reverting to the legacy mode of operation (where the msc*clk is on whenever the client clock is enabled.

Offset: 0xf009 │ Byte Offset: 0x3c024 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00xxxxxxxxxxx0000)

Bit	Reset	Description
17	0x0	G2_RCLK_OVERRIDE
16	0x0	G2_WCLK_OVERRIDE
3	DISABLE	G2_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	G2_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	G2_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	G2_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

14.10.11 G2SB_SWITCH_TIMEOUT_WCOAL_G2_0

Write Coalescing Time-Out Register

Note: Write coalescing is no longer supported by the MCCIF clients. Registers are retained for software compatibility but are not used by the hardware.

Offset: 0xf00a │ Byte Offset: 0x3c028 │ Read/Write: R/W │ Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	G2DW_WCOAL_TMVAL

14.10.12 G2SB_SWITCH_MCCIF_G2PR_HYST_0

Memory Client Hysteresis Control Register

Note: Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 0xf00b │ Byte Offset: 0x3c02c │ Read/Write: R/W │ Reset: 0xcf08ff06 (0b11001111000010001111111100000110)

Bit	Reset	Description
31	ENABLE	CBR_G2PR2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBR_G2PR2MC_HYST_REQ_TH
27:24	0xf	CBR_G2PR2MC_HYST_TM
23:16	0x8	CBR_G2PR2MC_DHYST_TH

Bit	Reset	Description
15:8	0xff	CBR_G2PR2MC_DHYST_TM
7:0	0x6	CBR_G2PR2MC_HYST_REQ_TM

14.10.13 G2SB_SWITCH_MCCIF_G2SR_HYST_0

Memory Client Hysteresis Control Register

Note: Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 0xf00c | Byte Offset: 0x3c030 | Read/Write: R/W | Reset: 0xcf08ff06 (0b11001111000010001111111100000110)

Bit	Reset	Description
31	ENABLE	CBR_G2SR2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBR_G2SR2MC_HYST_REQ_TH
27:24	0xf	CBR_G2SR2MC_HYST_TM
23:16	0x8	CBR_G2SR2MC_DHYST_TH
15:8	0xff	CBR_G2SR2MC_DHYST_TM
7:0	0x6	CBR_G2SR2MC_HYST_REQ_TM

14.10.14 G2SB_SWITCH_MCCIF_G2DR_HYST_0

Memory Client Hysteresis Control Register

Note: Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 0xf00d | Byte Offset: 0x3c034 | Read/Write: R/W | Reset: 0xcf04ff06 (0b11001111000001001111111100000110)

Bit	Reset	Description
31	ENABLE	CSR_G2DR2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CSR_G2DR2MC_HYST_REQ_TH
27:24	0xf	CSR_G2DR2MC_HYST_TM
23:16	0x4	CSR_G2DR2MC_DHYST_TH
15:8	0xff	CSR_G2DR2MC_DHYST_TM
7:0	0x6	CSR_G2DR2MC_HYST_REQ_TM

14.10.15 G2SB_SWITCH_MCCIF_G2DW_HYST_0

Memory Client Hysteresis Control Register

Note: Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 0xf00e │ Byte Offset: 0x3c038 │ Read/Write: R/W │ Reset: 0xc0000030 (0b1100xxxxxxxxxxxxxxxx000000110000)

Bit	Reset	Description
31	ENABLE	CCW_G2DW2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CCW_G2DW2MC_HYST_REQ_TH
11:0	0x30	CCW_G2DW2MC_HYST_REQ_TM

15.0 GR3D

15.1 Introduction

GR3D is the Tegra® GPU, a 3D graphics acceleration block, capable of a broad range of graphics and computing functions.

This section is not intended to be a programming guide to GR3D, or even a complete register description, but is provided solely to document which registers contain memory addresses. This allows software to be implemented which checks if the memory addresses being programmed are within valid ranges.

15.2 Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

The 3D unit can be instructed to access memory directly. To prevent using the 3D unit unintentionally, software might want to control access to the registers in 3D that contain memory addresses.

15.2.1 AR3D_IDX_ATTRIBUTE_0

Attribute Address and Format

This array sets the base address for each attribute.

This is an array of 16 identical register entries; the register fields below apply to each entry. There are 16 entries, where n is an integer from 0 through 15.

Offset: 0x100 + (n*2) | Byte Offset: 0x400 + (n*8) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxx0xxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ATTR_BASE: Base address of attribute array

15.2.2 AR3D_IDX_INDEX_BASE_0

Index Array Base

This register sets the index base for DrawElements commands.

Offset: 0x121 | Byte Offset: 0x484 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	INDEX_BASE: Base address of the index array for DrawElements.

15.2.3 AR3D_QR_ZTAG_ADDR_0

Offset: 0x415 | Byte Offset: 0x1054 | Read/Write: R/W | Reset: 0xXXXXXXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:6	X	BASE_ADDRESS: Start of the Z tag surface. This field defines the base address in 64-byte units/alignment.

15.2.4 AR3D_QR_CTAG_ADDR_0

Offset: 0x417 | Byte Offset: 0x105c | Read/Write: R/W | Reset: 0xXXXXXXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:6	X	BASE_ADDRESS: Start of the C tag surface. This field defines the base address in 64-byte units/alignment.

15.2.5 AR3D_QR_CZ_ADDR_0

Offset: 0x419 | Byte Offset: 0x1064 | Read/Write: R/W | Reset: 0xXXXXXXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:6	X	BASE_ADDRESS: Start of the CZ tag surface. This field defines the base address in 64-byte units/alignment.

15.2.6 AR3D_QR_FRAGMENT_PIPE_REGISTER_ADDR_0

This is an array of 4 identical register entries; the register fields below apply to each entry.

Offset: 0x420..0x423 | Byte Offset: 0x1080..0x108c | Read/Write: R/W | Reset: 0x000X0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
17	X	READ
16	X	INCR
11:0	X	ADDR

15.2.7 AR3D_TEX_TEXADDR_0

TEXADDR contains the base address for the texture surfaces.

This is an array of 16 identical register entries; the register fields below apply to each entry.

Offset: 0x710..0x71f | Byte Offset: 0x1c40..0x1c7c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BASE_ADDRESS: This surface base address should be 16-byte aligned.

15.2.8 AR3D_DW_MEMORY_OUTPUT_ADDRESS_0

Offset: 0x904 | Byte Offset: 0x2410 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BASE_ADDRESS

15.2.9 AR3D_GLOBAL_SURFADDR_0

SURFADDR -- Base Address for Frame Buffer Surfaces

This is an array of 16 identical register entries; the register fields below apply to each entry.

Offset: 0xe00..0xe0f | Byte Offset: 0x3800..0x383c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BASE_ADDRESS

15.2.10 AR3D_GLOBAL_SPILLSURFADDR_0

Base Address of a Pixel Spill Surface

Offset: 0xe2a | Byte Offset: 0x38a8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BASE_ADDRESS

15.2.11 AR3D_GLOBAL_SURFOVERADDR_0

Overlap Buffer

This is an array of 16 identical register entries; the register fields below apply to each entry.

Offset: 0xe30..0xe3f | Byte Offset: 0x38c0..0x38fc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BASE_ADDRESS

15.2.12 AR3D_GLOBAL_SAMP01SURFADDR_0

Surface Address for Multisample Sample 0 and 1 Buffers

This is an array of 16 identical register entries; the register fields below apply to each entry.

Offset: 0xe50..0xe5f | Byte Offset: 0x3940..0x397c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BASE_ADDRESS

15.2.13 AR3D_GLOBAL_SAMP23SURFADDR_0

Surface Address for Multisample Sample 2 and 3 Buffers

This is an array of 16 identical register entries; the register fields below apply to each entry.

Offset: 0xe60..0xe6f | Byte Offset: 0x3980..0x39bc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	BASE_ADDRESS

16.0 ENCODER PRE-PROCESSOR (EPP)

The Encoder Pre-Processor (EPP) is an engine used for pixel manipulation, usually in conjunction with GR2D, the 2D graphics engine.

This section is not intended to be a programming guide to EPP, as it is expected that NVIDIA supplied drivers will always be used with it. However, the register interface is documented to aid with understanding those drivers.

16.1 Capabilities

The EPP receives input stream or surface data from GR2D, the 2D graphics engine (or VI, the video/camera input path, but this path is not currently used).

The EPP receives input data on a 24-bit data bus with the format compatible to CSI output format. The input data formats are in 6 possible formats as far as EPP is concerned:

- 24-bit YUV444 (Y in bits 23:16, U in bits 15:8, V in bits 7:0)
- 24-bit RGB888 (R in bits 23:16, G in bits 15:8, B in bits 7:0)
- 1-byte Raw in bits 7:0
- 2-byte Raw in bits 15:0
- MIPI CSI YUV420 8-bit legacy
- MIPI CSI YUV420 8-bit

EPP supports various output data formats to memory: planar YUV, non-planar YUV, RGB, and Raw formats.

Supported YUV Planar Output Formats

a. YUV420P: planar YUV420 (3-plane YUV420)

Y-plane: YYYY...

YYYY...

U-plane: UU...

V-plane: VV...

b. YUV422P: planar YUV422 (3-plane YUV422)

Y-plane: YYYY...

U-plane: UU...

V-plane: VV...

c. YUV422R: planar YUV422 rotated (3-plane YUV422 rotated)

Y-plane: YYYY...

YYYY...

U-plane: UUUU...

V-plane: VVVV...

d. YUV420SP: semi-planar YUV420 (2-plane YUV420)

Y-plane: YYYY...

YYYY...

UV-plane: UVUV...

e. YUV422SP: semi-planar YUV422 (2-plane YUV422)

Y-plane: YYYY...

UV-plane: UVUV...

f. YUV422SPR: semi-planar YUV422 rotated (2-plane YUV422 rotated)

Y-plane: YY...

YY...

UV-plane: UVUV...

Supported YUV and RGB 32-bit Non-Planar Output Formats

This table, and the ones that follow it, show the specific bit mappings. The characters YUV and RGB refer to the corresponding fields of each color space; and A refers to an alpha value. A single row of digits below those characters shows the bit mapping of the sub-field corresponding to the letter above. For the fields with two rows of digits, the first row shows the ordering of multiple sub-fields; e.g., format ‘b’ contains two Y fields, so 0 and 1 show the mappings of each of those.

	LSB--------------------------MSB [0:31]
a. YUVV444	VVVVVVVVUUUUUUUUYYYYYYYYAAAAAAAA
	01234567012345670123456701234567
b. YUV422NPVYUY	VVVVVVVVYYYYYYYYUUUUUUUUYYYYYYYY
	00000000000000000000000011111111
	01234567012345670123456701234567
c. YUV422NPYVYU	YYYYYYYYVVVVVVVVYYYYYYYYUUUUUUUU
	00000000000000001111111100000000
	01234567012345670123456701234567
d. YUV422NPUYVY	UUUUUUUUYYYYYYYYVVVVVVVVYYYYYYYY
	00000000000000000000000011111111
	01234567012345670123456701234567
e. YUV422NPYUYV	YYYYYYYYUUUUUUUUYYYYYYYYVVVVVVVV
	11111111000000000000000000000000
	01234567012345670123456701234567
f. B8G8R8A8	BBBBBBBBGGGGGGGGRRRRRRRRAAAAAAAA
	01234567012345670123456701234567
g. R8G8B8A8	RRRRRRRRGGGGGGGGBBBBBBBBAAAAAAAA
	01234567012345670123456701234567
h. A8B8G8R8	AAAAAAAABBBBBBBBGGGGGGGGRRRRRRRR
	01234567012345670123456701234567
i. A8R8G8B8	AAAAAAAARRRRRRRRGGGGGGGGBBBBBBBB
	01234567012345670123456701234567

Supported RGB 16-bit Non-Planar Output Formats

```
                                    LSB -------- MSB [0:15]
a. B5G6R5                           BBBBBGGGGGGRRRRR
                                    0123401234501234
b. R5G6R5                           RRRRRGGGGGGBBBBB
                                    0123401234501234
c. B5G5R5A1                         BBBBBGGGGGRRRRRA
                                    0123401234012340
d. R5G5B5A1                         RRRRRGGGGGBBBBBA
                                    0123401234012340
e. A1B5G5R5                         ABBBBBGGGGGRRRRR
                                    0012340123401234
f. A1R5G5B5                         ARRRRRGGGGGBBBBB
                                    0012340123401234
g. B4G4R4A4                         BBBBGGGGRRRRAAAA
                                    0123012301230123
h. R4G4B4A4                         RRRRGGGGBBBBAAAA
                                    0123012301230123
i. A4B4G4R4                         AAAABBBBGGGGRRRR
                                    0123012301230123
j. A4R4G4B4                         AAAARRRRGGGGBBBB
                                    0123012301230123
```

Supported Non-Planar Raw Output Formats

```
                                    LSB--MSB [0:7]
a. RAW8 (1-byte raw)                DDDDDDDD
                                    01234567


                                    LSB -------- MSB [0:15]
b. RAW16 (2-byte raw)               DDDDDDDDDDDDDDDD
                                    0123456789111111
                                              012345
```

For all output formats that require alpha (A), the alpha value is programmed by the host and the same programmed value is used for all output pixels.

If EPP output processing is stalled due to a stall from the memory controllers or due to an output trigger stall, the EPP will propagate the stall condition to its input FIFO. If the input FIFO becomes full, then the EPP will not accept any more input data.

This stall may cause input data stream to be dropped (thrown away) if the input stream cannot be stalled or if the input stream comes from the host via VI, then VI will stall host input stream processing.

16.2 Reset Sequence for EPP

1. Raise VI at the end of the EPP frame.

   If the first_output of VI is connected to the EPP, then RAISE_FRAME_1_VECTOR should be sent to register NV_VI_RAISE_VIP_FRAME_FIRST_OUTPUT_0 (offset 0x3e). If the second output of VI is connected to the EPP, then RAISE_FRAME_2_VECTOR should be sent to register NV_VI_RAISE_VIP_FRAME_SECOND_OUTPUT_0 (offset 0x40).
2. Disable the VI output to the EPP by setting bits 4:2 to 0 of register NV_VI_VI_CORE_CONTROL_0 (offset 0x3)
3. Raise the EPP at end of frame from host, NV_EPP_RAISE_FRAME_0 (offset 0x1a)
4. Read the NV_EPP_CTXW_0 (0ffset 0) register, and save the value as EPP_CTXSW_DATA

5. Write 0 to NV_MC_CLIENT_CTRL_0 (offset 0x18), bit 4 to block EPP requests inside the MC.
6. Reset the EPP by write 0 to bit 6 of NV_HOST1X_ASYNC_RSTREG_0 (offset 0x14). Do a read-modify-write.
7. Write 0 to NV_MC_CLIENT_HOTRESETN_0 (offset 0x19), bit 4 to clear EPP requests blocked inside the MC.
8. Poll on NV_MC_EPP_ORRC_0 (offset 0x1e), MC_EPP_ORRC_0_EPP_OUTREQCNT_RANGE field (0xFF), till it is zero.
9. Write 1 to NV_MC_CLIENT_HOTRESETN_0, bit 4 to disable the clear of EPP requests inside the MC.
10. Write 1 to NV_MC_CLIENT_CTRL_0, bit 4 to unblock the EPP request inside the MC.
11. Bring the EPP out of reset by writing 1 to bit 6 of NV_HOST1X_ASYNC_RSTREG_0 (offset 0x14). Do a read-modify-write.
12. Program NV_EPP_CTXSW_0 (0x0) with the previously saved EPP_CTXSW_DATA.
13. Program the EPP registers
14. Enable the encoder
15. Enable the VI EPP output by programming bits 4:2 to the desired format of NV_VI_VI_CORE_CONTROL_0 (offset 0x3)

16.3 EPP Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

16.3.1 EPP_INCR_SYNCPT_0

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	INDX: syncpt index value

16.3.2 EPP_INCR_SYNCPT_CNTRL_0

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when the FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when the FIFOs are full, the client host interface will be stalled.

Bit	Reset	Description
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

16.3.3 EPP_INCR_SYNCPT_ERROR_0

Offset: 0x2 │ Byte Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

16.3.4 EPP_EPP_SYNCPT_DEST_0

This is for COND_FIFO_DEPTH (ignored for COND 0). COND_TRIG_MODE applies to COND 1..x (ignored for 0,1).

Offset: 0x8 │ Byte Offset: 0x20 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx01)

Bit	Reset	Description
1	0x0	MPE
0	0x1	HOST

16.3.5 EPP_CTXSW_0

Context Switch Register. Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Context switch works like this:

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts.

Another way to avoid context switch interrupts is to set the AUTO_ACK bit. This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x9 │ Byte Offset: 0x24 │ Read/Write: R/W │ Reset: 0xf000f800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK

Bit	R/W	Reset	Description
9:0	RW	0x0	CURR_CLASS: Current working class

16.3.6 EPP_INTSTATUS_0

Interrupt Status. This register reflects the status of all pending interrupts, which is valid as long as the interrupt is not cleared even if the interrupt is masked. A pending interrupt can be cleared by writing a '1' to the corresponding interrupt status bit in this register.

Offset: 0xa │ Byte Offset: 0x28 │ Read/Write: R/W │ Reset: 0xff000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31:24	RO	X	LAST_BUFFER_INDEX: Last buffer index status. This indicates the index of the previous (last) buffer written to memory by the EPP.
8	RW	X	SHORT_FRAME_INT: Short Frame Interrupt Status. If enabled, an interrupt is generated when a shorter than expected frame is detected. Input stream end-of-frame flag and OUTPUT_FRAME_SIZE are used to determine this condition. 0 = NOTPENDING: interrupt not pending 1 = PENDING: interrupt pending
2	RW	X	BUFFER_END_INT: Output Buffer End Interrupt Status. If enabled, an interrupt is generated every time an end of buffer is reached. Typically, this is determined when OB0_V_SIZE count expires and after all buffer data is written to memory. Note that the buffer may not be completely filled for first and last buffer of each frame. This interrupt should be used together with LAST_BUFFER_INDEX status. 0 = NOTPENDING: interrupt not pending 1 = PENDING: interrupt pending
1	RW	X	FRAME_END_INT: Frame End Interrupt Status. If enabled, an interrupt is generated every time the FRAME_HEIGHT count expires after all frame data is written to memory. 0 = NOTPENDING: interrupt not pending 1 = PENDING: interrupt pending
0	RW	X	CTXSW_INT: Context Switch Interrupt Status (this is cleared on write). 0 = NOTPENDING: interrupt not pending 1 = PENDING: interrupt pending

16.3.7 EPP_EPP_CONTROL_0

EPP Control Register. This specifies processing control.

Offset: 0xb │ Byte Offset: 0x2c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx0xx00000000000000000000)

Bit	Reset	Description
24	0x0	SW_FLOW_CONTROL: enable software flow control default is disabled. 0 = DISABLE 1 = ENABLE
21:18	0x0	OUTPUT_FORMAT_EXT: Output data format extension. This specifies output data format together with OUTPUT_FORMAT. RGB formats are specified from lsb to msb. 0 = EXT0 : YUV420: planar YUV422: non-planar if OUTPUT_PLANAR=DISABLE, planar if OUTPUT_PLANAR=ENABLE. YUV422R: planar YUV444: AYUV444 non-planar if OUTPUT_PLANAR=DISABLE RGB888: B8G8R8A8 if OUTPUT_PLANAR=DISABLE, R8G8B8A8 if OUTPUT_PLANAR=ENABLE. RGBRAW: B5G6R5 if OUTPUT_PLANAR=DISABLE, R5G6B5 if OUTPUT_PLANAR=ENABLE. BAYERRAW: 2-byte bayer/raw 1 = EXT1 : YUV420: semi-planar YUV422: semi-planar if OUTPUT_PLANAR=ENABLE. YUV422R: semi-planar RGB888: A8B8G8R8 if OUTPUT_PLANAR=DISABLE, A8R8G8B8 if OUTPUT_PLANAR=ENABLE. RGBRAW: A1B5G5R5 if OUTPUT_PLANAR=DISABLE, A1R5G5B5 if OUTPUT_PLANAR=ENABLE. BAYERRAW: 1-byte bayer/raw 2 = EXT2 : YUV420: planar, CSI YUV420 legacy input RGBRAW: B5G5R5A1 if OUTPUT_PLANAR=DISABLE, R5G5B5A1 if OUTPUT_PLANAR=ENABLE. 3 = EXT3 : YUV420: semi-planar, CSI YUV420 legacy input RGBRAW: A4B4G4R4 if OUTPUT_PLANAR=DISABLE, A4R4G4B4 if OUTPUT_PLANAR=ENABLE.

Bit	Reset	Description
		4 = EXT4 : YUV420: planar, CSI YUV420 input. RGBRAW: B4G4R4A4 if OUTPUT_PLANAR=DISABLE, R4G4B4A4 if OUTPUT_PLANAR=ENABLE. 5 = EXT5 : YUV420: semi-planar, CSI YUV420 input.
17	0x0	ENABLE_DUP: Pixel duplication at start and end of lines. This enables duplication of the first pixel and last pixel of each line to 128-bit boundary, if there is unused space in the 128-bit memory word that contains the first and last pixel. This may be enabled if the output of the EPP is used as input of JPEG encoder. The JPEGE encoder can duplicate lines to ensure JPEGE has a full MCU to encode. If the EPP Is used as an input to the MPEG encoder, the output height of the EPP has to be set to multiples of 16 because the MPEG encoder does not duplicate lines. This is only applicable for planar YUV output formats (OUTPUT_PLANAR=ENABLE and output Format!=RAWRGB and output Format!=RGB888). 0 = DISABLE : First and last pixel duplication is disabled. 1 = ENABLE : First and last pixel duplication is enabled for each line.
16	0x0	DMA_ENABLE: DMA enable. Software should program this bit to a 1 every time DMA is enabled and buffer configuration changes This bit is now sent out as init for all the host counters and logic. This enables EPP trigger at end of each buffer to be sent to Read DMA. The Read DMA must be properly programmed to fetch EPP buffers in memory upon receiving these triggers. 0 = DISABLE: EPP trigger to Read DMA is disabled 1 = ENABLE: EPP trigger to Read DMA is enabled
15	0x0	CHROMA_SIGN: Chroma data sign. This indicates chroma data format. 0 = UNSIGNED: unsigned chroma data, Cb/Cr 1 = SIGNED: signed chroma data, U/V, in 2's complement
14:12	0x0	OUTPUT_FORMAT: Output data format. This specifies output data format together with OUTPUT_FORMAT_EXT. 0 = YUV420: YUV420 planar/semi-planar. ENABLE_422 must be set to ENABLE for this output format. ENABLE_420 must be set to ENABLE if the input format is not MIPI CSI YUV420 formats and must be set to DISABLE if the input format is MIPI CSI YUV420 formats. 1 = YUV422: YUV422 non-planar/planar/semi-planar. If YUV422 planar/semi-planar format is selected and XY swap is enabled, then the output will be written to memory in YUV422R planar/semi-planar format correspondingly. YUV422 non-planar format cannot be used with XY swap enabled. ENABLE_422 must be set to ENABLE and ENABLE_420 must be set to DISABLE for this output format. 2 = YUV422R: YUV422R planar/semi-planar. If YUV422R planar/semi-planar format is selected and XY swap is enabled, then the output will be written to memory in YUV422 planar/semi-planar format correspondingly. ENABLE_422 must be set to DISABLE and ENABLE_420 must be set to ENABLE for this output format. 3 = YUV444 : AYUV444 non-planar if OUTPUT_PLANAR=DISABLE. 4 = RGB888 : ARGB888 5 = RGBRAW : RGBRAW 6 = BAYERRAW : BAYERRAW
11	0x0	CHROMA_FILTER_422: Chroma YUV444 to YUV422 conversion. This controls chroma reduction when YUV444 to YUV422 conversion is enabled. This is effective only when ENABLE_422 is set to ENABLE. 0 = DROP: Even chroma pixels are dropped. This may be used if the incoming YUV444 stream has chroma pixel duplication of odd and even pixels. 1 = FILTER: Horizontal chroma filtering is applied for YUV444 to YUV422 conversion.
10	0x0	OUTPUT_PLANAR: Output planar format. This should be set to ENABLE if output format is YUV planar or semi-planar and should be DISABLE if output format is YUV non-planar. For RGB output data formats, this bit has alternate function of swapping R and B. 0 = DISABLE: For YUV output formats, output is YUV non-planar format. For RGB output formats, R is in upper bits and B is in lower bits. 1 = ENABLE: Output is YUV planar/semi-planar format. For RGB output formats, B is in upper bits and R is in lower bits.
9:8	0x0	OUTPUT_NP_FORMAT: Non-planar YUV422 output format. This selects output format when output format is set to YUV422 non-planar. 0 = UYVY: UY0VY1, where U is LSB and Y1 is MSB 1 = VYUY: VY0UY1, where V is LSB and Y1 is MSB 2 = YUYV: Y0UY1V, where Y0 is LSB and V is MSB 3 = YVYU: Y0VY1U, where Y0 is LSB and U is MSB

Bit	Reset	Description
7:6	0x0	CHROMA_FILTER_420: Chroma YUV422 to YUV420 conversion. This controls chroma reduction when YUV422 to YUV420 conversion is enabled. This is effective only when ENABLE_420 is set to ENABLE. 0 = DROP: Even chroma lines are dropped. 1 = AVERAGE: Averaging is done for each pair (odd and even) of chroma lines.
5	0x0	ENABLE_PP: Luma PreProcess filter enable. This controls luma pre-encoding filter for YUV output formats including YUV420 when input format is YUV420 CSI format. Enabling the luma pre-encoding filter will reduce noise in the input stream and may result in more compact encoding. 0 = DISABLE: Luma pre-encoding filter is disabled. 1 = ENABLE: Luma pre-encoding filter is enabled.
4	0x0	ENABLE_420: YUV422 to YUV420 enable. If set to ENABLE, chroma data is vertically reduced by half. YUV422 to YUV420 conversion must be enabled when OUTPUT_FORMAT is YUV420 and input format is not MIPI CSI YUV420. YUV422 to YUV420 conversion must also be enabled when OUTPUT_FORMAT is YUV422R. The vertical chroma reduction method is controlled by CHROMA_FILTER_420 field. 0 = DISABLE: YUV422 to YUV420 conversion is disabled. 1 = ENABLE: YUV422 to YUV420 conversion is enabled.
3	0x0	ENABLE_422: YUV444 to YUV422 enable. If set to ENABLE, chroma data is horizontally reduced by half. YUV444 to YUV422 conversion must be enabled when OUTPUT_FORMAT is YUV422 or YUV420. YUV444 to YUV422 conversion must be disabled when OUTPUT_FORMAT is YUV422R. The horizontal chroma reduction method is controlled by CHROMA_FILTER_422 field. 0 = DISABLE: YUV444 to YUV422 conversion is disabled. 1 = ENABLE: YUV444 to YUV422 conversion is enabled.
2	0x0	ENABLE_CC: Color Space Converter enable. This enables RGB to YUV color space converter and should be enabled only for RGB input format if the output format is YUV. If enabled, the output of the color space converter is YUV444 therefore conversion to YUV422 or YUV420 may be needed also. 0 = DISABLE: RGB input format is not converted to YUV. 1 = ENABLE: RGB input format is converted to YUV.
1:0	0x0	INPUT_SOURCE: Input source. 0 = VI: Input source from VI. 1 = SB: Input source from StretchBLT (2D). 2 = DISPLAY: Input source from DISPLAY. 3 = DISPLAYB : Input source from DISPLAY B

16.3.8 EPP_OUTPUT_FRAME_SIZE_0

Captured, Input Frame Size

This specifies the width and height of the input frame with respect to the size of the active area to be processed by the EPP. For some planar and semi-planar YUV output formats, there are 2:1 ratio of luma vs chroma pixels horizontally and/or vertically. In this case, output frame size specifies luma plane size and should be programmed to even values if the corresponding chroma plane dimension is half of the luma plane dimension.

Offset: 0xc │ Byte Offset: 0x30 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	FRAME_HEIGHT: Frame height in lines (minimum 1 line).
15:0	0x0	FRAME_WIDTH: Frame width in pixels (minimum 16 pixels).

16.3.9 EPP_INPUT_FRAME_AOI_0

Input Frame Area of Interest

This specifies the position of the first pixel in the input frame to be processed by the EPP. Together with the output frame size, this parameter defines the active size of the input surface.

Offset: 0xd │ Byte Offset: 0x34 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	FRAME_VERT_OFFSET: Vertical offset in term of line position (minimum 0).
15:0	0x0	FRAME_HORI_OFFSET: Horizontal offset in term of pixel position (minimum 0).

16.3.10 EPP_OUTPUT_SCAN_DIR_0

Output Scan Direction

This register can be used to program the output frame orientation as follows:

XY_SWAP	VERT_DIR	HORI_DIR	Output frame orientation
0	0	0	Normal (same as input frame orientation)
0	0	1	H flip (mirror on vertical axis)
0	1	0	V flip (mirror on horizontal axis)
0	1	1	180-degree rotation
1	0	0	Reserved, do not use.
1	0	1	Reserved, do not use.
1	1	0	Reserved, do not use.
1	1	1	Reserved, do not use.

Notes:

a. XY swap feature is deprecated and must not be used.

b. Output buffer start address should be programmed in consideration of the scan directions.

OB0_Start_Address = OBSA + hsd * (width * bpp - 1) + vsd * (height - 1) * line_stride, where line_stride >= ((width * bpp) + 15) & 0xFFFFFFF0).

In the above formulae, OBSA is the start address of output buffer 0, hsd and vsd are H/V scan direction, bpp is bytes per pixel, width and height are the sizes of input image in pixels, and line_stride is the line stride of output buffer in bytes.

c. For YUV422 non-planar, since chroma data is shared by multiple pixels, XY swap CANNOT be performed.

Offset: 0xe │ Byte Offset: 0x38 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	XY_SWAP: Reserved, set to DISABLE (zero) 0 = DISABLE
1	0x0	VERT_DIR: Vertical scan direction. 0 = INCREASE: Increasing address. 1 = DECREASE: Decreasing address.
0	0x0	HORI_DIR: Horizontal scan direction. 0 = INCREASE: Increasing address. 1 = DECREASE: Decreasing address.

Output buffer start addresses define the start address of the first buffer (buffer index 0) and taking into account the output orientation. Note that since output buffer start addresses depend on output orientation, therefore it must be reprogrammed when output orientation is changed.

All addresses must be on pixel boundaries.

Output format is either Planar or Non-planar format. For non-planar output format, only luma buffer is used.

For semi-planar output formats, only 2 planes are used. Chroma data is interleaved in a single plane. So for semi-planar outputs, only OB0_START_ADDRESS_U is used to specify chroma buffer start address. Enough memory should be reserved to store U and V data for the chroma buffer.

16.3.11 EPP_OB0_BASE_ADDRESS_Y_0

Offset: 0x10 | Byte Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OB0_BASE_ADDRESS_Y: OB0 Y BASE address.

16.3.12 EPP_OB0_BASE_ADDRESS_U_0

Output buffer 0 U base address. This is used to specify the U buffer base address for planar output format. This is also used for the semi-planar chroma buffer base address. This is not used for the non-planar output format.

Offset: 0x12 | Byte Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OB0_BASE_ADDRESS_U: OB0 U start address.

16.3.13 EPP_OB0_START_ADDRESS_V_0

Output buffer 0 V start address. This is used to specify the V buffer start address for the planar output format. This is not used for semi-planar or non-planar output format.

Offset: 0x13 | Byte Offset: 0x4c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OB0_START_ADDRESS_V: OB0 V start address.

16.3.14 EPP_OB0_BASE_ADDRESS_V_0

Output buffer 0 V base address. This is used to specify the V buffer base address for the planar output format. This is not used for semi-planar or non-planar output format.

Offset: 0x14 | Byte Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	OB0_BASE_ADDRESS_V: OB0 V start address.

16.3.15 EPP_OB0_XY_OFFSET_LUMA_0

XY Offset to first pixel in the image. The value is expressed in pixels. For non-rotated surfaces, this points to the TOP-LEFT of the image.

Offset: 0x15 | Byte Offset: 0x54 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	START_Y_LUMA: Y offset in pixels for the first pixel in the image in the Y buffer.
15:0	0x0	START_X_LUMA: X offset in pixels for the first pixel written in the image in the Y buffer

16.3.16 EPP_OB0_XY_OFFSET_CHROMA_0

XY Offset to first pixel in the image. The value is expressed in pixels. For non-rotated surfaces, this points to the TOP-LEFT of the image.

Offset: 0x16 | Byte Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	START_Y_CHROMA: Y offset in pixels for the first pixel in the image in the U and V buffer.
15:0	0x0	START_X_CHROMA: X offset in pixels for the first pixel written in the image in the U and V buffer. For YUV422R or YUV420P with Chroma Averaging on START_X_CHROMA must be aligned to a 16 byte boundary.

16.3.17 EPP_OB0_SIZE_0

Output buffer 0 size. This specifies the number of output buffers and the vertical size of each buffer. Note that horizontal size of each buffer is implied from OB_LINE_STRIDE.

Offset: 0x17 | Byte Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
28:16	X	OB0_V_SIZE: Output buffer vertical size. This specifies the number of lines. In the case of xyswap, the vertical size should be programmed as the vertical size before xyswap.
7:0	0x0	OB0_COUNT: Output buffer count. This specifies the number of buffers in output buffer set 0.

16.3.18 EPP_OB0_LINE_STRIDE_L_0

Output buffer line stride. Line stride should be programmed taking into account XY swap. If XY swap is enabled, line stride should be programmed as after xyswap.

Offset: 0x18 | Byte Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	OB0_LINE_STRIDE_C: Output buffer chroma line stride. This parameter must be programmed as a 16-byte multiple so the 4 lsbs must be set to zeros.
15:0	0x0	OB0_LINE_STRIDE_L: Output buffer luma line stride. This parameter must be programmed as a 16-byte multiple so the 4 lsbs must be set to zeros.

Color Space Converter related registers are CSC_RGB2Y_COEFF, CSC_RGB2U_COEFF, CSC_RGB2V_COEFF and CSC_YOFFSET_COEFF.

The recommended coefficient values differ based on the type of conversion either RGB to ITU-R BT.601 Y/Cb/Cr or RGB to ITU-R BT.709 Y/Cb/Cr is used. For both the type of conversions, R,G,B values are expected to be in the range of [0,255] and after Color Space Conversion operation, Y value is in the range of [16,235] with Y-Offset of 16, Cb and Cr are in the range of [16,240] with offset of 128. The Offset of 128 for Cb and Cr are hard-coded in the design.

16.3.19 EPP_CSC_RGB2Y_COEFF_0

RGB to Y Coefficients

The coefficient for R component CSC_R2Y_COEFF is represented as Unsigned U0.8 (8 bits), the coefficient for G component CSC_G2Y_COEFF is represented as Unsigned U1.8 (9 bits), and the coefficient for B component CSC_B2Y_COEFF is represented as Unsigned U0.8 (8 bits).

For RGB to ITU-R BT.601 Y/Cb/Cr conversion, the recommended coefficient values are:

[CSC_R2Y_COEFF CSC_G2Y_COEFF CSC_B2Y_COEFF] = [0.256788 0.504129 0.097906]

The coefficients are represented with an 8-bit fraction (U1.8 or U0.8). Therefore for a dynamic range of 256 (8 bits), the programmed coefficient values are [66 129 25] = [42h 81h 19h]

For RGB to ITU-R BT.709 Y/Cb/Cr conversion, the recommended coefficient values are:

[CSC_R2Y_COEFF CSC_G2Y_COEFF CSC_B2Y_COEFF] = [0.182586 0.614231 0.062007]

The coefficients are represented with an 8-bit fraction (U1.8 or U0.8). Therefore for a dynamic range of 256 (8 bits), the programmed coefficient values are [47 157 16] = [2fh 9dh 10h]

Offset: 0x1b │ Byte Offset: 0x6c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24:17	0x0	CSC_B2Y_COEFF: Coefficient for B to Y component.
16:8	0x0	CSC_G2Y_COEFF: Coefficient for G to Y component.
7:0	0x0	CSC_R2Y_COEFF: Coefficient for R to Y component.

16.3.20 EPP_CSC_RGB2U_COEFF_0

RGB to U Coefficients

The coefficient for R component CSC_R2U_COEFF is represented as Sign Magnitude S0.8 (9-bits), the coefficient for G component CSC_G2U_COEFF is represented as Sign Magnitude S0.8 (9-bits), the coefficient for B component CSC_B2U_COEFF is represented as Sign Magnitude S0.8 (9-bits).

The most significant bit (9th bit) is either set to 0 or 1 based on the coefficient value being positive or negative, respectively.

For RGB to ITU-R BT.601 Y/Cb/Cr conversion, the recommended coefficient values are:

[CSC_R2U_COEFF CSC_G2U_COEFF CSC_B2U_COEFF] = [-0.148223 -0.290993 0.439216]

The coefficients are represented with an 8-bit fraction (S0.8). Therefore for a dynamic range of 256 (8 bits), the programmed coefficient values are [-38 -74 112] = [-26h -4ah 70h] = [126h 14ah 70h]

For RGB to ITU-R BT.709 Y/Cb/Cr conversion, the recommended coefficient values are:

[CSC_R2U_COEFF CSC_G2U_COEFF CSC_B2U_COEFF] = [-0.100644 -0.338572 0.439216]

The coefficients are represented with an 8-bit fraction (S0.8). Therefore for a dynamic range of 256 (8 bits), the programmed coefficient values are [-26 -87 112] = [-1ah -57h 70h] = [11ah 157h 70h]

Offset: 0x1c │ Byte Offset: 0x70 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxx00000000000000000000000)

Bit	Reset	Description
26:18	0x0	CSC_B2U_COEFF: Coefficient for B to U component.
17:9	0x0	CSC_G2U_COEFF: Coefficient for G to U component.
8:0	0x0	CSC_R2U_COEFF: Coefficient for R to U component.

16.3.21 EPP_CSC_RGB2V_COEFF_0

RGB to V Coefficients

The coefficient for R component CSC_R2V_COEFF is represented as Sign Magnitude S0.8 (9 bits), the coefficient for G component CSC_G2V_COEFF is represented as Sign Magnitude S0.8 (9 bits), and the coefficient for B component CSC_B2V_COEFF is represented as Sign Magnitude S0.8 (9 bits).

The most significant bit (9th bit) is either set to 0 or 1 based on the coefficient value being positive or negative, respectively.

For RGB to ITU-R BT.601 Y/Cb/Cr conversion, the coefficient values recommended are:

[CSC_R2V_COEFF CSC_G2V_COEFF CSC_B2V_COEFF] = [0.439216 -0.367788 -0.040274]

The coefficients are represented with 8-bit fraction (S0.8) therefore for a dynamic range of 256 (8-bits), the programmed coefficient values are [112 -94 -18] = [70h -5eh -12h] = [70h 15eh 112h]

For RGB to ITU-R BT.709 Y/Cb/Cr conversion, the coefficient values recommended are:

[CSC_R2V_COEFF CSC_G2V_COEFF CSC_B2V_COEFF] = [0.439216 -0.398942 -0.071427]

The coefficients are represented with an 8-bit fraction (S0.8). Therefore for a dynamic range of 256 (8 bits), the programmed coefficient values are [112 -102 -10] = [70h -66h -0ah] = [70h 166h 10ah]

Offset: 0x1d │ Byte Offset: 0x74 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxx00000000000000000000000000)

Bit	Reset	Description
26:18	0x0	CSC_B2V_COEFF: Coefficient for B to V component.
17:9	0x0	CSC_G2V_COEFF: Coefficient for G to V component.
8:0	0x0	CSC_R2V_COEFF: Coefficient for R to V component.

16.3.22 EPP_CSC_YOFFSET_COEFF_0

Y Offset for the Color Space Conversion

Offset: 0x1e │ Byte Offset: 0x78 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000) │ Default: 0x00000010

Bit	Reset	SW Default	Description
7:0	0x0	0x10	CSC_YOFF_COEFF: Coefficient for Y Offset.

There are two filters in EPP -- luma pre-processing filter and chroma 444 to 420 filter. Each filter is designed to be 7 tab, LPF.

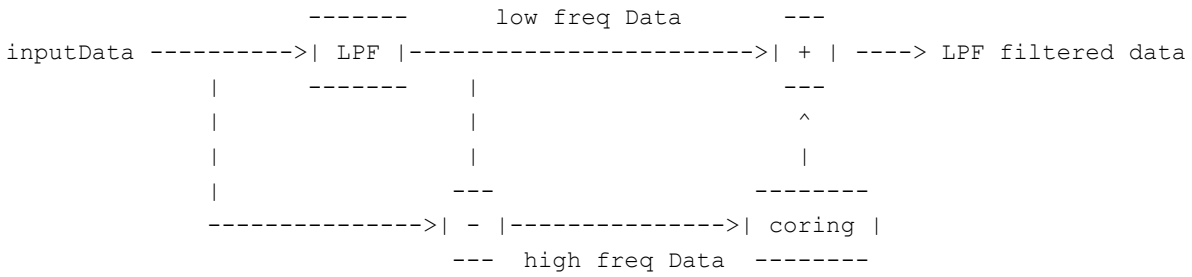
The recommended coefficients for luma pre-processing filter are: (0 1 4 6 4 1 0)/16. The filter coefficients are symmetric; therefore only four coefficients are programmed.

The recommended coefficient base is 16 and programmed to 4, in terms of powers of 2. The sum of all coefficients should not exceed the coefficient base value.

The recommended coefficients for chroma 444-to-422 filter are: (-1 0 9 16 9 0 -1)/64. The filter coefficients are symmetric, therefore only four coefficients are programmed.

The recommended coefficient base is 64 and programmed to 6, in terms of powers of 2. The sum of all coefficients should not exceed the coefficient base value.

The basic concept for this filter is:



16.3.23 EPP_FILTER_BOUND_0

Filter coring bound. This specifies minimum/maximum values for filter coring for both the luma pre-processing filter and the chroma 444-to-422 filter.

Offset: 0x1f │ Byte Offset: 0x7c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000) │ Default: 0xff00ff00

Bit	Reset	SW Default	Description
31:24	0x0	0xff	CHROMA_THRESHOLD_HIGH: Chroma high bound
23:16	0x0	_NONE_	CHROMA_THRESHOLD_LOW: Chroma low bound
15:8	0x0	0xff	LUMA_THRESHOLD_HIGH: Luma high bound
7:0	0x0	_NONE_	LUMA_THRESHOLD_LOW: Luma low bound

16.3.24 EPP_FILTER_BASE_0

Filter coefficient base

Offset: 0x20 │ Byte Offset: 0x80 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000xxxxxxxx00000000) │ Default: 0x00040004

Bit	Reset	SW Default	Description
23:16	0x0	0x4	CHROMA_BASE: Chroma 444-to-422 filter coefficient base. This is a positive value programmed in terms of power of 2.
7:0	0x0	0x4	PP_BASE: Luma pre-processing filter coefficient base. This is a positive value programmed in terms of power of 2.

16.3.25 EPP_PP_FILTER_COEF_0

Luma pre-processing filter coefficients. This 7-tap filter is symmetric therefore only four coefficients are programmed. The sum of all 7 coefficients should not exceed the filter base value.

Recommended luma pre-processing filter is a 5-tap filter (0,1,4,6,4,1,0)/16.

An alternate recommendation is a 7-tap filter (1,6,15,20,15,6,1)/64.

Offset: 0x21 │ Byte Offset: 0x84 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxx00000000000000000000) │ Default: 0x00031020

Bit	Reset	Default	Description
18:13	0x0	0x18	PP_COEF_3: Luma pre-processing filter coefficients 4. This is an unsigned value 0 to 63
12:8	0x0	0x10	PP_COEF_2: Luma pre-processing filter coefficients 3, 5. This is a unsigned value 0 to 31.
7:3	0x0	0x4	PP_COEF_1: Luma pre-processing filter coefficients 2, 6. This is a signed value -16 to +15.
2:0	0x0	0x0	PP_COEF_0: Luma pre-processing filter coefficients 1, 7. This is a signed value -4 to +3.

16.3.26 EPP_CHROMA_FILTER_COEF_0

Chroma 444-to-422 filter coefficients. This 7-tap filter is symmetric therefore only four coefficients are programmed. The sum of all 7 coefficients should not exceed the filter base value.

Recommended chroma 444-to-422 filter is a 5-tap filter (0,1,4,6,4,1,0)/16.

An alternate recommendation is a 7-tap filter (-1,0,9,16,9,0,-1)/64.

Offset: 0x22 | Byte Offset: 0x88 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx00000000000000000000) | Default: 0x00031020

Bit	Reset	SW Default	Description
18:13	0x0	0x18	CHROMA_COEF_3: Chroma 444-to-422 filter coefficients 4. This is an unsigned value 0 to 63
12:8	0x0	0x10	CHROMA_COEF_2: Chroma 444-to-422 filter coefficients 3, 5. This is a unsigned value 0 to 31.
7:3	0x0	0x4	CHROMA_COEF_1: Chroma 444-to-422 filter coefficients 2, 6. This is a signed value -16 to +15.
2:0	0x0	0x0	CHROMA_COEF_0: Chroma 444-to-422 filter coefficients 1, 7. This is a signed value -4 to +3.

16.3.27 EPP_ALFA_VALUE_0

Output alpha value. This is an appended alpha value for RGB and/or YUV444 output data formats that requires alpha. If the data format requires less than 8 bits of alpha, then the necessary least significant bits are used.

Offset: 0x23 | Byte Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	ALFA: Output alpha value.

16.3.28 EPP_U_LINE_BUFFER_ADDR_0

Chroma line buffer for U and V data. Chroma buffer needs to be allocated whenever ENABLE_420 is set to ENABLE, and CHROMA_FILTER_420 is set to AVERAGE.

Offset: 0x24 | Byte Offset: 0x90 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	U_LINE_BUFFER_ADDR: Chroma line buffer for U & V data. This must be specified on a 16-byte boundary (the four LSBs must be set to zeros). The size of this line buffer must be sufficient to store one line of U & V.

16.3.29 EPP_V_LINE_BUFFER_ADDR_0

Note: This register is no longer used.

Offset: 0x25 | Byte Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	V_LINE_BUFFER_ADDR: Reserved.

EPP may accept a raise from VI via the same input data bus from VI or two types of raises from host.

A raise from VI is returned when all preceding input data have been processed and written to memory.

Two types of raises maybe sent from the host:

- Raise Buffer
- Raise Frame

For raises from host, EPP tags last write for each buffer and each frame when sending it to memory client and then it waits for the tag to be returned. Typically, raise buffer will be acknowledged when the next end of buffer tag is returned by the memory write client.

Similarly, typically, raise frame will be acknowledged when the next end of frame tag is returned by the memory write client.

If raise buffer or raise frame is issued during vertical blank time (between end-of-frame marker of previous frame and start-of-frame of next frame) then an option bit is added to either acknowledge the raise immediately or return the raise when the next end of buffer or end of frame tag is returned by the memory write client correspondingly.

When one or more event that returns data to host are pending, priority will be assigned as follows:

1. Context Switch Acknowledgement
2. Raise from VI
3. Raise Buffer
4. Raise Frame
5. Refcount or register read

16.3.30 EPP_RAISE_BUFFER_0

Raise Buffer (end of buffer raise). The end of buffer event is generated when the last data of the output buffer is written to memory. This is independent of output trigger so the end of buffer raise acknowledge is returned even though there may be an output trigger stall pending.

Offset: 0x26 │ Byte Offset: 0x98 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxx00000xxxxxxxxxx00000)

Bit	Reset	Description
19:16	0x0	CHANNEL_ID_BUFFER: 4-bit channel ID which will be returned when the end-of-buffer event occurs after raise buffer is issued.
15	0x0	RAISE_BUFFER_VBLANK: Raise buffer control during V Blank. This specifies when to return raise buffer if it is issued during vertical blank time. 0 = IMMEDIATE: Raise buffer is acknowledged immediately if it is issued during vertical blank time. 1 = DELAYED: Raise buffer is acknowledged at the end of the first buffer of the next frame if it is issued during vertical blank time.
4:0	0x0	RAISE_VECTOR_BUFFER: 5-bit raise vector

16.3.31 EPP_RAISE_FRAME_0

Raise Frame (end of frame raise). The end of frame event is generated when the last data of the output buffer is written to memory. This is independent of output trigger so the end of buffer raise acknowledge is returned even though there may be an output trigger stall pending.

Offset: 0x27 │ Byte Offset: 0x9c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxx00000xxxxxxxxxx00000)

Bit	Reset	Description
19:16	0x0	CHANNEL_ID_FRAME: 4-bit channel ID which will be returned when the end-of-frame event occurs after raise frame is issued.
15	0x0	RAISE_FRAME_VBLANK: Raise frame control during V Blank. This specifies when to return raise frame if it is issued during vertical blank time. 0 = IMMEDIATE: Raise frame is acknowledged immediately if it is issued during vertical blank time. 1 = DELAYED: Raise frame is acknowledged at the end of the next frame if it is issued during vertical blank time.
4:0	0x0	RAISE_VECTOR_FRAME: 5-bit raise vector

17.0 KEYBOARD CONTROLLER

The Keyboard Controller (KBC) is used to interface a maximum of 19 pins with an external keypad. The number of pins used in the KBC depends on the external keypad row and column count.

A keyboard controller for a keypad matrix implemented completely in software can result in significant overhead. The KBC lowers the burden on software by supporting keypad scan, debounce and wake-up on any key press in hardware. It also reduces power consumption in keypad associated operations. The remaining pins can be used as GPIO. If no external keypad is connected all the pins can be used as GPIO. The KBC module is a slave on the APB bus and a part of the APBDEV module.

The typical software scanning flow is shown in Figure 40. Most of the tasks in the flowchart can be done in hardware, off-loading them from software. The three main operations done by the KBC are:

- Keypad scan
- Debounce
- Control state machine (SM)

In the presence of a KBC, the software flowchart for keyboard control is reduced to what is shown in Figure 41. The software only needs to configure the keyboard controller and serve the interrupts. This automatically reduces the power consumption related to keypad operations.

Features

The KBC supports:

- Keypad matrix size of up to 11X8.
- Wake-up on interrupt mode on unmasked pins
- Restricts active rows to be mapped starting from row0 in order
- Continuous polling mode
- Multi-key press support (dependency on keypad hardware)
- Joystick (16 ms key-press speed)

The keypad matrix provides ghost key protection in case of three or more simultaneous key presses. The system has a power ON/OFF key. The handling of this key is a little different from the rest of the keys. The KBC is not aware of this difference and treats it as any other key; different modules in the Tegra® 4 series processor (e.g., Always ON) monitor this key when the device is powered off.

Row 0 should always be enabled. Rows remain inactivate during the wakeup mode when all the columns of that row are masked. Rows that are enabled must be contiguous. For example, if we need to enable three rows, then they must be rows 0, 1, and 2. Enabling rows 0, 1, and 3, for example, is an invalid combination since the rows are not contiguous.

All the wakeup keys must be on the same column [n], other non-wakeup keys should not be part of the column [n] to avoid spurious wake-up interrupts.

Figure 40: Typical Software Scanning Flow

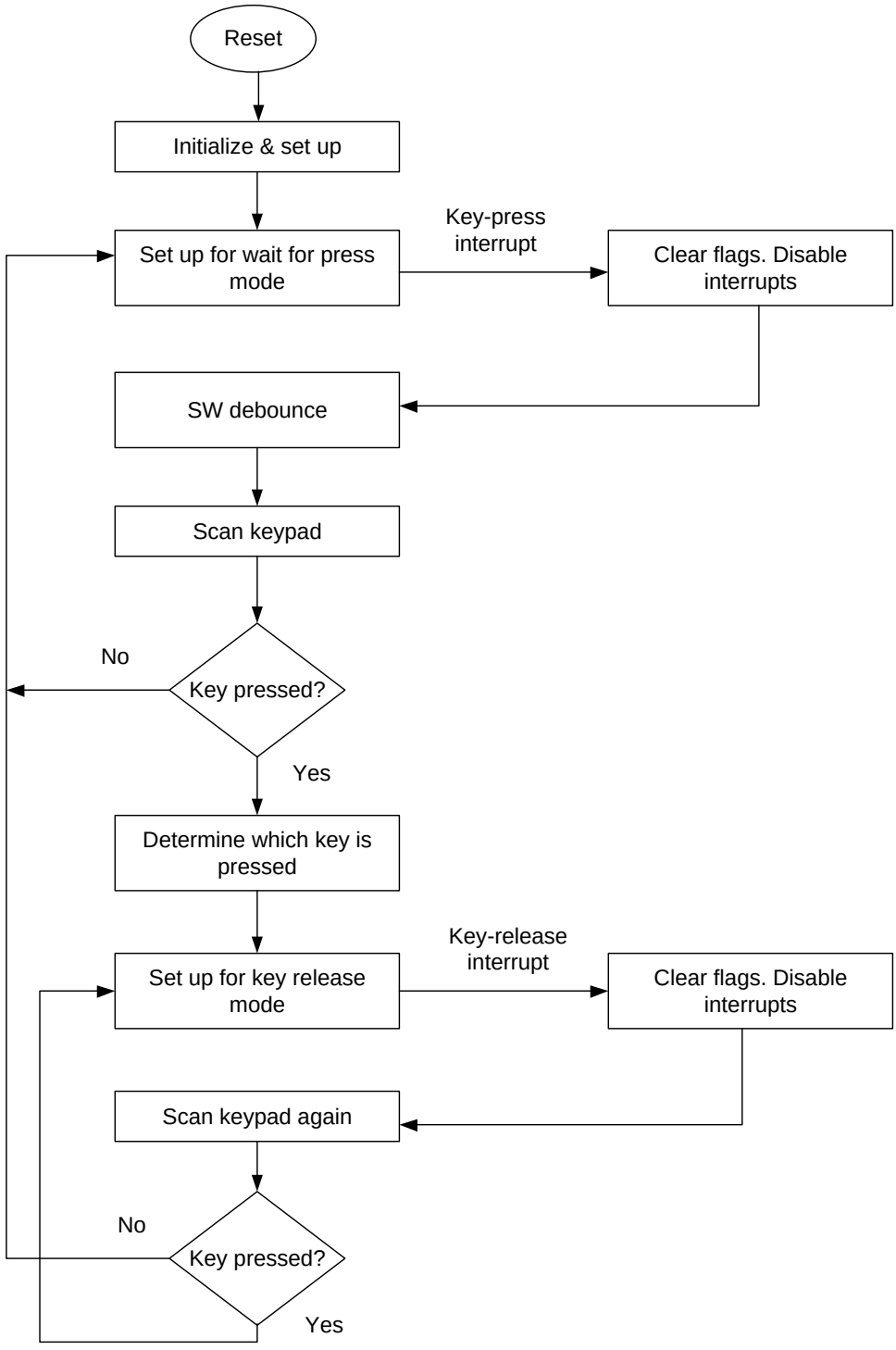


Figure 41: Software Flow with Hardware Keyboard Controller

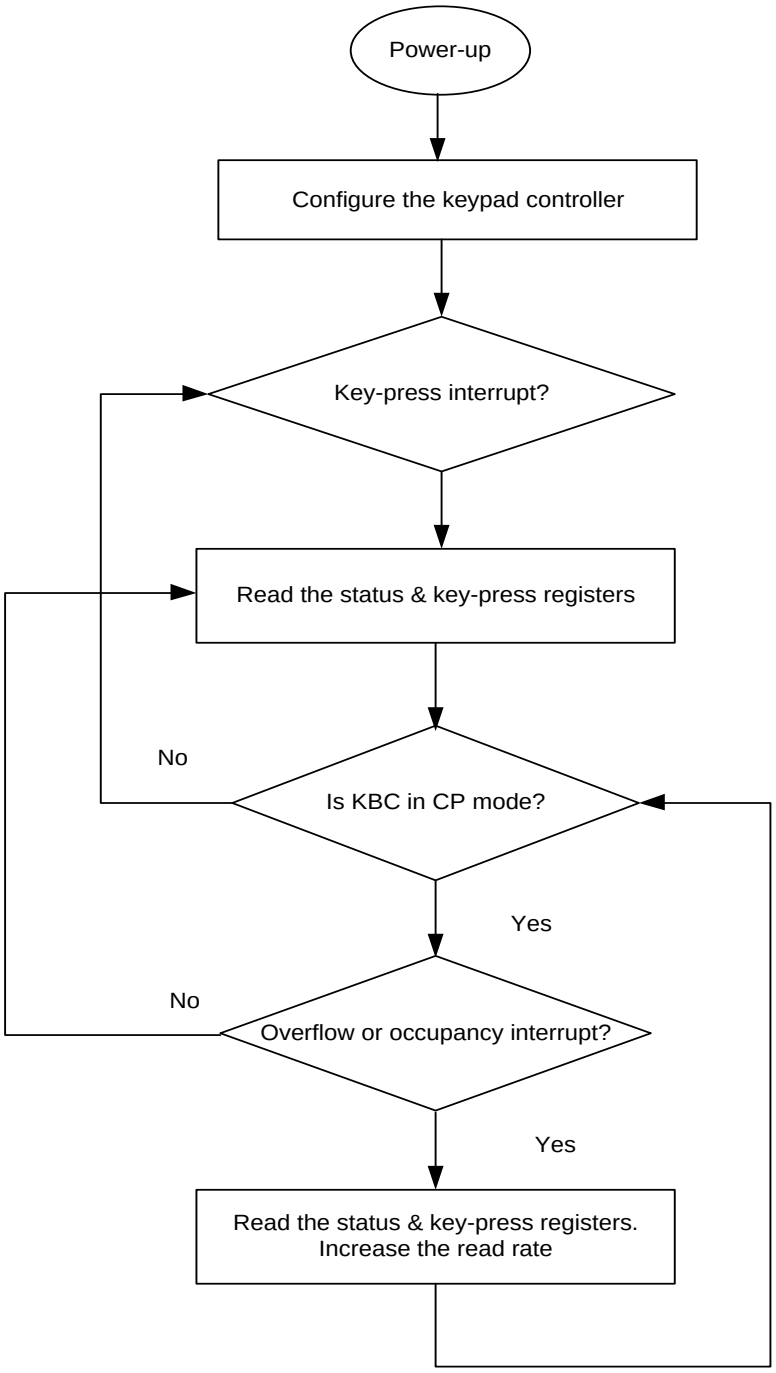
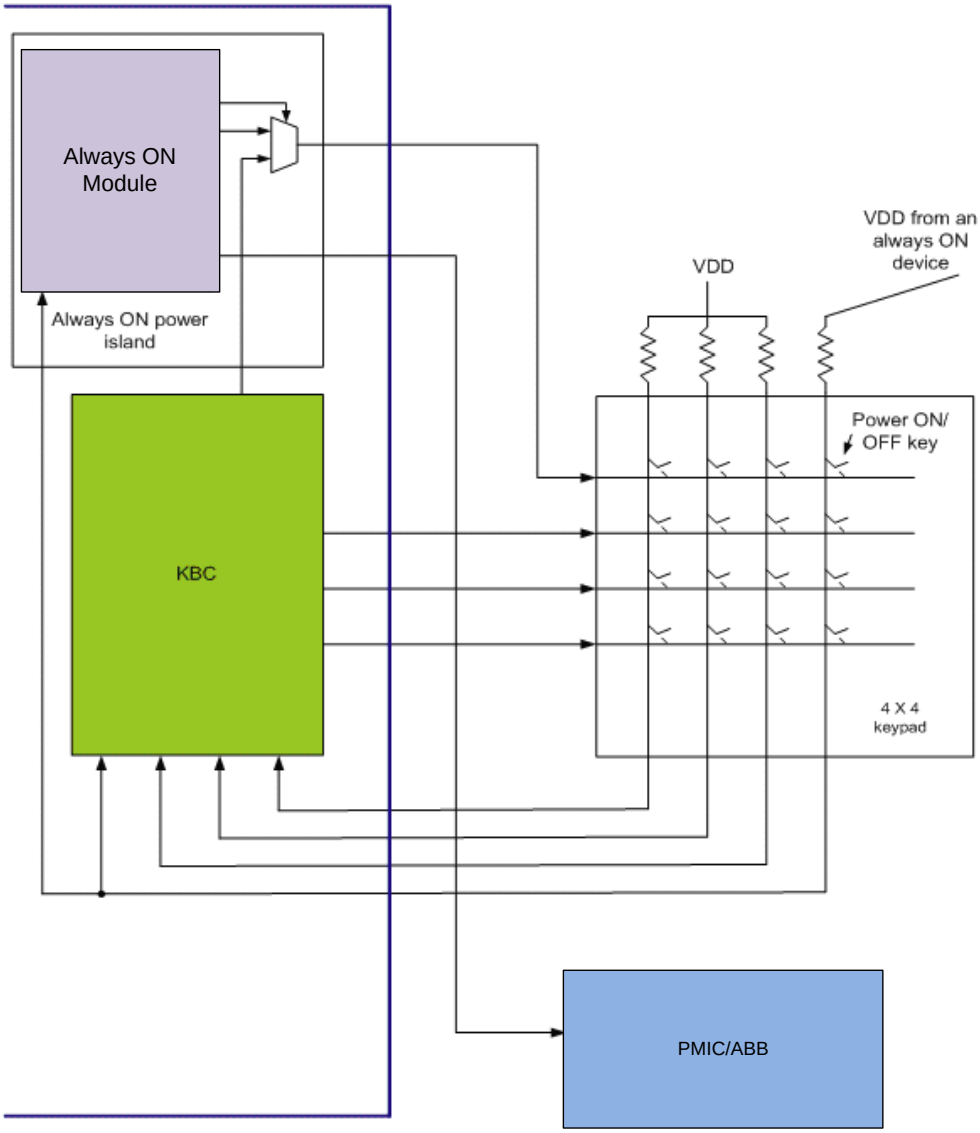


Figure 42: Power ON/OFF Key Example

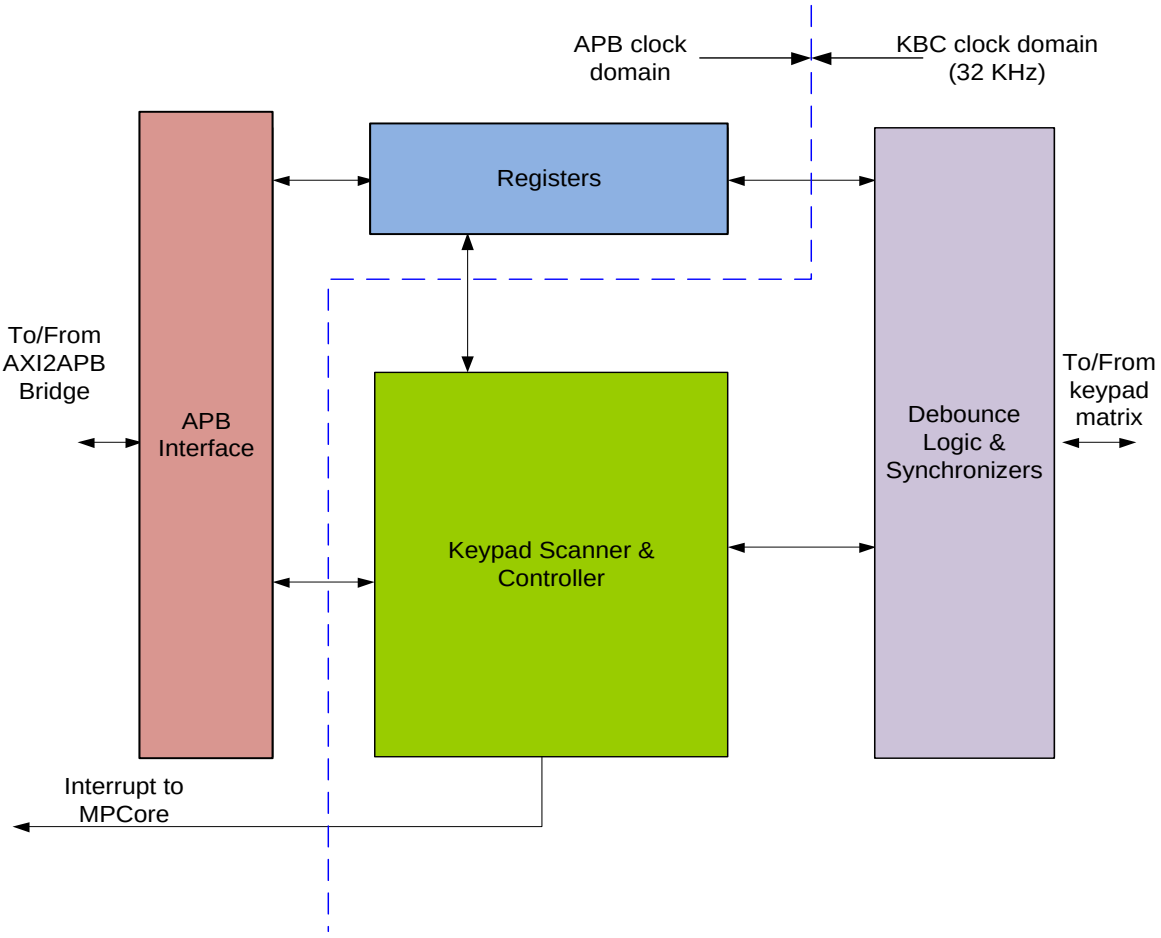


Clocking

- APB clock for APB interface and register sub-module
- KBC clock for keypad scanner and controller (32 KHz)

17.1 Functionality

Figure 43: KBC Functional Block Diagram



The keyboard controller is enabled when the enable bit is set, and some pins are configured for interfacing with keypad matrix. The pins which are not interfaced to the keypad matrix can be used as GPIO pins. To configure the KBC, the following registers should be programmed:

- APBDEV_KBC_ROW_CFG
- APBDEV_KBC_INT
- APBDEV_KBC_COL_CFG
- APBDEV_KBC_CONTROL

The KBC supports a keypad of maximum size 11 X 8 (row X column). All the pins interfacing with the keypad use debounce logic to detect a valid key press (filter out glitches).

Initially, the KBC is in wake-up state on key press (WuKP). In this mode, it waits for any key press. All rows are driven low to detect any key press in this state. Once a key is pressed, the KBC generates an optional key-press interrupt (KP_INT_STATUS in APBDEV_KBC_INT register), goes to continuous polling (CP) state and starts scanning after INIT_DLY_VAL (in APBDEV_KBC_INIT_DLY register) cycles.

In the CP state, the delay between two consecutive full scans is RPT_DLY_VAL (in APBDEV_KBC_RPT_DLY register) cycles. In the CP state, it scans all the rows one-by-one by driving one row low while other rows are configured as inputs so they are driven "Z". Then all the columns are sampled for each row scan.

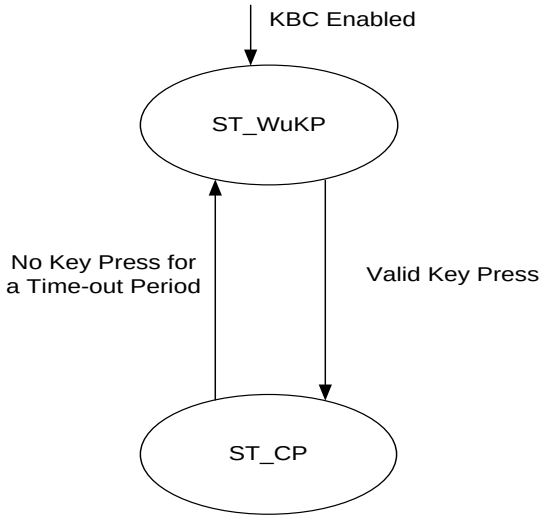
The driving of rows and sampling of columns is based on the debounce cycle. The KBC cycle is 32 KHz (time period = 31.25 microseconds). So if the debounce count is 4, one row scan takes 125 microseconds. One complete scan of all rows (16 maximum) takes 2 milliseconds. In one second there can be 500 complete scans (if RPT_DLY_VAL = 0).

The KBC supports up to eight simultaneous key presses per complete scan. These eight key numbers are stored in the APBDEV_KBC_KP_ENT0 and APBDEV_KBC_KP_ENT1 registers. The FIFO pointer increments on the second register entry register (APBDEV_KBC_KP_ENT1) read. Software should always read the two entry registers consecutively. It should first read APBDEV_KBC_KP_ENT0 and then read APBDEV_KBC_KP_ENT1.

To avoid more than one wrong key press, software should check the AV_FIFO_CNT value in APBDEV_KBC_INT register. The entry registers give new key-press information only when the AV_FIFO_CNT value is non-zero. A zero value indicates that there are no new key presses after the last read. To indicate whether an entry is valid or not, a valid bit is associated with every entry in the two entry registers.

Software should read the entry register fast enough in CP state to avoid overflow. To avoid an overflow in case of slow reads, the first level warning indicator can be programmed in FIFO_TH_CNT field of the APBDEV_KBC_CONTROL register. An optional interrupt (FIFO_CNT_INT_STATUS in APBDEV_KBC_INT register) can be generated when the FIFO occupancy count reaches this value. In case of overflow an optional interrupt (FIFO_OVF_INT_STATUS in APBDEV_KBC_INT register) is generated and software will have two options of treating the new entries. It can either drop them or overwrite them. This bit is FIFO_MODE in APBDEV_KBC_CONTROL register. The figure below shows KBC control of the SM.

Figure 44: KBC Control Logic



17.2 LP0 Entry/Exit Procedure

- The KBC shall be in Wake/Interrupt mode, i.e., the default state where all rows are driving zero and non-scan mode. To immediately abort the scan operation and enter wake mode, program a zero to the KBC register APBDEV_KBC_TO_CNT_0.
- Confirm the KBC enters wake mode, i.e., APBDEV_KBC_INT_0[KBC_ST_STATUS] shall be zero, restore the APBDEV_KBC_TO_CNT_0 with the appropriate value.
- Initiate LP0 Entry sequence.
- Exit the LP0 mode on KBC event.

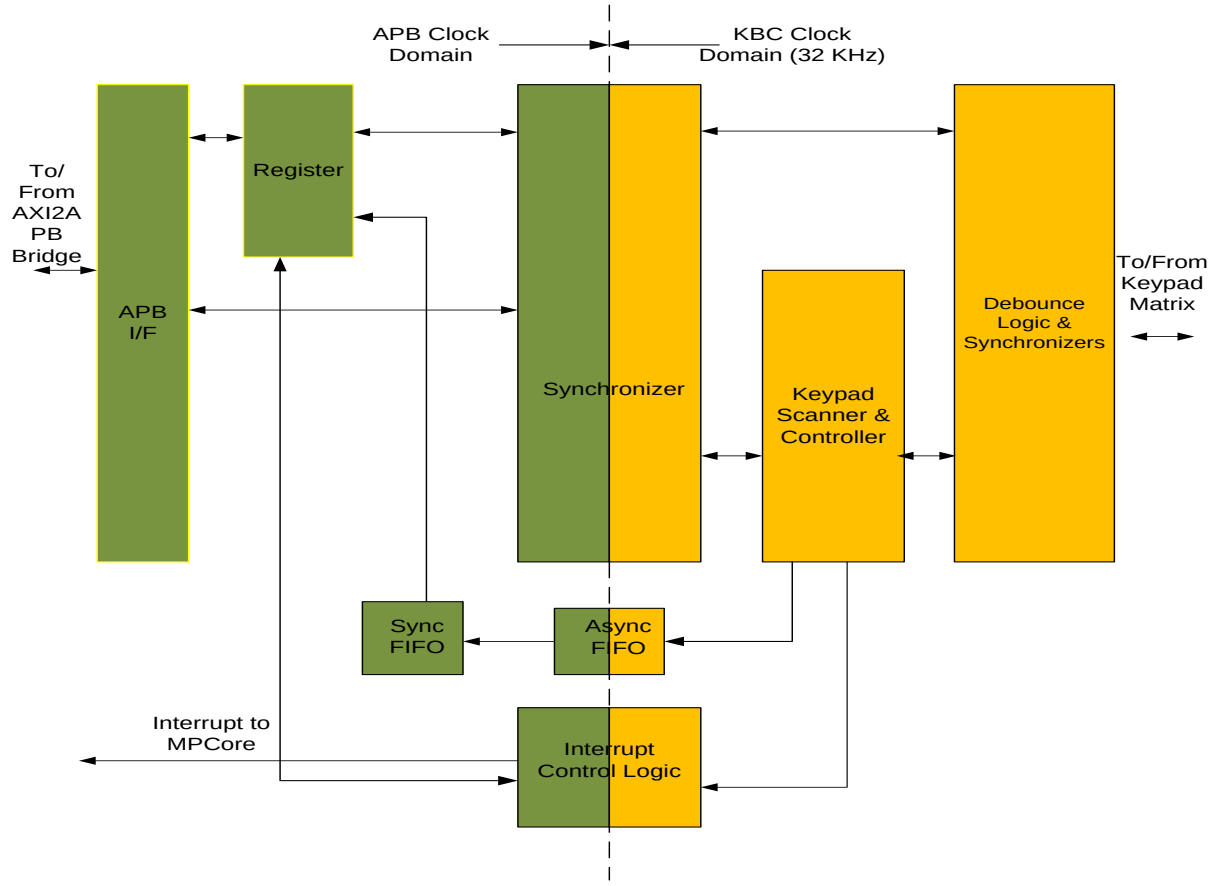
17.3 Micro-Architecture

17.3.1 Keypad Matrix

The next figure shows the top level diagram of the KBC with all the sub-modules. The Tegra 4 series KBC controller supports an 11x8 keypad matrix (11 rows and 8 columns). Up to eight simultaneous key presses can be stored per full scan (row scanning). Storage for key-press events is as follows: an asynchronous FIFO stores 1 key-press event, a synchronous FIFO stores 8 events, and a register stores 1 event.

A minimum of two rows must always be enabled.

Figure 45: KBC Block Diagram



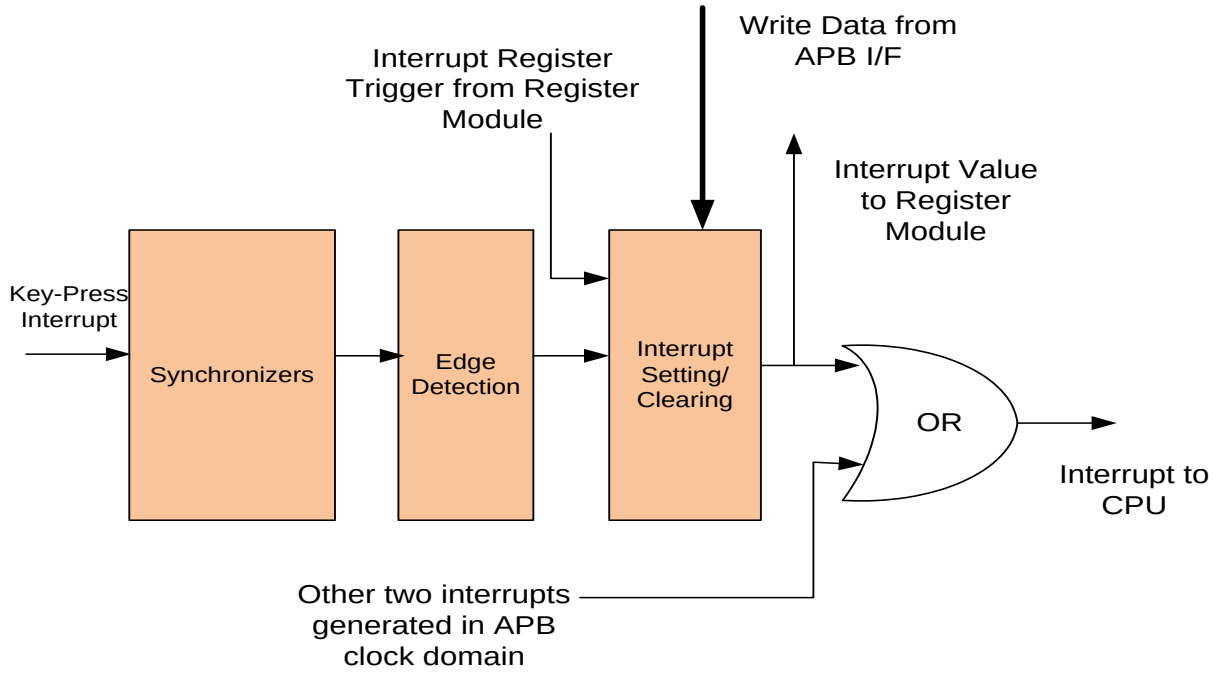
17.3.2 Synchronizer

Register sync from one synchronization from one clock domain to another is achieved through generation of a ‘valid’ signal. This valid signal is synchronized instead of all the data bits of the register. Unless an acknowledgment is received (for this ‘valid’) from the receiving side, the valid request is not deasserted and data (going to the receiving side) is not changed. The acknowledgment is deasserted only when valid request is deasserted. If there is no valid asserted for a register, the value to be transmitted is compared with the previous sent value and if they are unequal, the valid is asserted and the new data transmitted. Otherwise, valid remains deasserted and data is not changed. The module “async_cross_c” is used for this synchronization.

17.3.3 Interrupt Control Logic

The key-press interrupt is generated on the KBC clock, which is much slower than the APB and MPCore clocks. The interrupt should be cleared on the APB clock so that the delay between an interrupt clear in a register bit to actual deassertion of the interrupt is not visible to the MPCore. To deassert the interrupt in the APB clock domain, it is first synchronized. On a positive edge of interrupt from keypad scanner the interrupt to the MPCore is asserted and it is cleared when the corresponding register bit is written with a ‘1’. The other two interrupts (FIFO threshold count and overflow) are generated in the APB clock domain.

Figure 46: Interrupt Control Logic

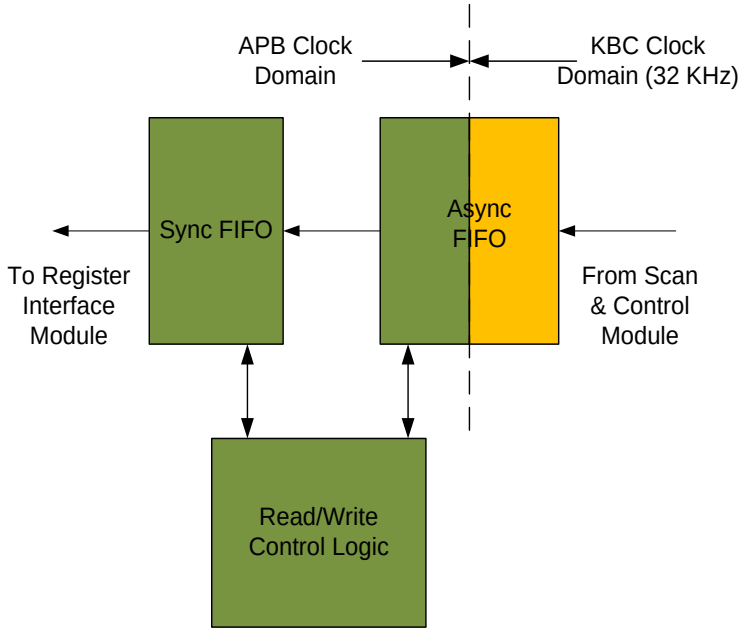


17.3.4 Keypad Scanner and Controller

This sub-module contains the main state machine. It controls the scanning, generates interrupts, identifies the keys which are pressed, and puts them into the appropriate registers. It also updates the KP_NEW_ENT* bits associated with all keys to indicate whether a key in the register is valid or not.

The software can program the KBC for two options in case of FIFO overflow. Either drop the new entries or overwrite the old entries with new entries. The design has two FIFOs: one single deep asynchronous FIFO and an eight deep synchronous FIFO. When the KBC is configured to overwrite on FIFO overflow, on getting an asynchronous FIFO non-empty signal when synchronous FIFO is full, a dummy read is done from synchronous FIFO. Then the asynchronous FIFO read value is written to

synchronous FIFO. This effectively overwrites the old entry. When the KBC is configured to drop new entries on FIFO overflow, this dummy read is not done which will effectively drop the entry.



17.3.5 Additional Interface Signals

kbc_interrrupt : An interrupt in the 32 KHz domain is sent to PMC and is used to wake up the system.

17.3.6 Pull-ups and Pull-downs in Pads

All the pads have pull-ups and pull-down bits which can be programmed by the software per your requirement(s). Contact your NVIDIA representative for details on the pull-up/pull-down registers.

Rows are always outputs to the Tegra 4 series processor and Columns are inputs. Row and Columns for the I/O can be programmed by software as needed. For Tegra 4 series processors the following I/O mapping is used:

ROW [15:0] --- gp3_pr [7:0], gp3_ps [2:0]

COLUMN [7:0] --- gp3_pq[7:0]

17.4 Programming Guidelines

The keyboard controller is enabled when the enable bit is set and some pins are configured for interfacing with the keypad matrix. The pins which are not interfaced to the keypad matrix can be used as GPIO pins.

To configure the KBC, the following registers should be programmed:

- APBDEV_KBC_ROW_CFG
- APBDEV_KBC_INT
- APBDEV_KBC_COL_CFG
- APBDEV_KBC_CONTROL.

The KBC supports a keypad of maximum size 11 X 8 (row X column). All the pins interfacing with the keypad use debounce logic to detect a valid key press (filter out glitches).

To avoid an overflow in case of slow reads, the first level warning indicator can be programmed in the FIFO_TH_CNT field of the APBDEV_KBC_CONTROL register.

In case of overflow, software will have two options of treating the new entries: either drop them or overwrite them. This bit is FIFO_MODE in the APBDEV_KBC_CONTROL register.

17.5 KBC Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

17.5.1 APBDEV_KBC_CONTROL_0

KBC control register. This register is the main control register. It should be configured last after all other settings are configured.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00010000 (0bxxxxxxxxxxxxxx00100000000000000000)

Bit	Reset	Description
18	0x0	FIFO_MODE: Selects the behavior in case of FIFO overflow. 0 = Drop the new detected key presses. 1 = Overwrite the new detected key presses
17:14	0x4	FIFO_TH_CNT: FIFO threshold count. Keeps the threshold FIFO occupancy count. If the FIFO reaches/crosses that count an optional interrupt will be raised. Should not be programmed as 0. N = Threshold occupancy count is N
13:4	0x0	DBC_CNT: Debounce count. This value sets the debounce FSM associated with each KBC input pin evaluate the input transitions. 0 = No debounce. N = N KBC clocks
3	0x0	FIFO_CNT_INT_EN: FIFO threshold count interrupt enable. Setting this bit will enable interrupts when FIFO occupancy reaches/crosses the value specified in FIFO_TH_CNT. 0 = Disable FIFO overflow interrupt. 1 = Enable FIFO overflow interrupt
2	0x0	FIFO_OVF_INT_EN: FIFO overflow interrupt enable. Setting this bit will enable interrupt on FIFO overflow
1	0x0	KP_INT_EN: Key-press interrupt enable. Setting this bit will enable interrupt on any key press 0 = DISABLE 1 = ENABLE
0	0x0	EN: Keyboard controller enable. Setting this bit will override the pins settings done in GPIO 0 = DISABLE 1 = ENABLE

17.5.2 APBDEV_KBC_INT_0

KBC Interrupt Status and Clear Register

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	R/W	Reset	Description
7:4	RO	X	AV_FIFO_CNT: FIFO occupancy count. Shows the number of unread registers.
3	RO	X	KBC_ST_STATUS: KBC status. 0 = WuKP (Wake-up on key press) interrupt mode. 1 = CP (Continuous polling) mode
2	RW	0x0	FIFO_CNT_INT_STATUS: FIFO threshold count interrupt status. Writing '1' to this bit will clear the interrupt. 0 = FIFO threshold count interrupt deasserted. 1 = FIFO threshold count interrupt asserted (read). 1 = Clear FIFO overflow interrupt (write)

Bit	R/W	Reset	Description
1	RW	0x0	FIFO_OVF_INT_STATUS: FIFO overflow interrupt status. Writing '1' to this bit will clear the interrupt. 0 = FIFO overflow interrupt deasserted. 1 = FIFO overflow interrupt asserted (read). 1 = Clear FIFO overflow interrupt (write)
0	RW	0x0	KP_INT_STATUS: Key-press interrupt status. Writing '1' to this bit will clear the interrupt. 0 = Key-press interrupt deasserted. 1 = Key-press interrupt asserted (read). 1 = Clear key-press interrupt (write)

17.5.3 APBDEV_KBC_ROW_CFG0_0

First row configuration register. This register setting will take precedence in case the same GPIO pin is configured in the column register also. It does the mapping between the GPIO pins and rows of the keypad matrix. It configures for GPIO pin numbers 0 to 5.

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000000000000000000000)

Bit	Reset	Description
29:26	0x0	GPIO_5_ROW_NUM: Mapping of GPIO pin #5 to row number. Valid only if GPIO_5_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
25	0x0	GPIO_5_ROW_EN: Indicates whether GPIO pin #5 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #5 in column configuration 0 = NOT_MAPPED 1 = MAPPED
24:21	0x0	GPIO_4_ROW_NUM: Mapping of GPIO pin #4 to row number. Valid only if GPIO_4_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
20	0x0	GPIO_4_ROW_EN: Indicates whether GPIO pin #4 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #4 in column configuration 0 = NOT_MAPPED 1 = MAPPED
19:16	0x0	GPIO_3_ROW_NUM: Mapping of GPIO pin #3 to row number. Valid only if GPIO_3_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
15	0x0	GPIO_3_ROW_EN: Indicates whether GPIO pin #3 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #3 in column configuration 0 = NOT_MAPPED 1 = MAPPED
14:11	0x0	GPIO_2_ROW_NUM: Mapping of GPIO pin #2 to row number. Valid only if GPIO_2_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
10	0x0	GPIO_2_ROW_EN: Indicates whether GPIO pin #2 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #2 in column configuration 0 = NOT_MAPPED 1 = MAPPED
9:6	0x0	GPIO_1_ROW_NUM: Mapping of GPIO pin #1 to row number. Valid only if GPIO_1_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
5	0x0	GPIO_1_ROW_EN: Indicates whether GPIO pin #1 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #1 in column configuration 0 = NOT_MAPPED 1 = MAPPED
4:1	0x0	GPIO_0_ROW_NUM: Mapping of GPIO pin #0 to row number. Valid only if GPIO_0_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15

Bit	Reset	Description
0	0x0	GPIO_0_ROW_EN: Indicates whether GPIO pin #0 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #0 in column configuration 0 = NOT_MAPPED 1 = MAPPED

17.5.4 APBDEV_KBC_ROW_CFG1_0

Second row configuration register. This register setting will take precedence in case the same GPIO pin is configured in the column register also. It does the mapping between the GPIO pins and rows of the keypad matrix. It configures for GPIO pin numbers 6 to 11.

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000000000000000000000)

Bit	Reset	Description
29:26	0x0	GPIO_11_ROW_NUM: Mapping of GPIO pin #11 to row number. Valid only if GPIO_11_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
25	0x0	GPIO_11_ROW_EN: Indicates whether GPIO pin #11 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #11 in column configuration 0 = NOT_MAPPED 1 = MAPPED
24:21	0x0	GPIO_10_ROW_NUM: Mapping of GPIO pin #10 to row number. Valid only if GPIO_10_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
20	0x0	GPIO_10_ROW_EN: Indicates whether GPIO pin #10 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #10 in column configuration 0 = NOT_MAPPED 1 = MAPPED
19:16	0x0	GPIO_9_ROW_NUM: Mapping of GPIO pin #9 to row number. Valid only if GPIO_9_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
15	0x0	GPIO_9_ROW_EN: Indicates whether GPIO pin #9 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #9 in column configuration 0 = NOT_MAPPED 1 = MAPPED
14:11	0x0	GPIO_8_ROW_NUM: Mapping of GPIO pin #8 to row number. Valid only if GPIO_8_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
10	0x0	GPIO_8_ROW_EN: Indicates whether GPIO pin #8 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #8 in column configuration 0 = NOT_MAPPED 1 = MAPPED
9:6	0x0	GPIO_7_ROW_NUM: Mapping of GPIO pin #7 to row number. Valid only if GPIO_7_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
5	0x0	GPIO_7_ROW_EN: Indicates whether GPIO pin #7 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #7 in column configuration 0 = NOT_MAPPED 1 = MAPPED
4:1	0x0	GPIO_6_ROW_NUM: Mapping of GPIO pin #6 to row number. Valid only if GPIO_6_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
0	0x0	GPIO_6_ROW_EN: Indicates whether GPIO pin #6 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #6 in column configuration 0 = NOT_MAPPED 1 = MAPPED

17.5.5 APBDEV_KBC_ROW_CFG2_0

Third row configuration register. This register setting will take precedence in case the same GPIO pin is configured in column register also. It does the mapping between the GPIO pins and rows of the keypad matrix. It configures for GPIO pin numbers 12 to 17.

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0bxx00000000000000000000000000000000)

Bit	Reset	Description
29:26	0x0	GPIO_17_ROW_NUM: Mapping of GPIO pin #17 to row number. Valid only if GPIO_17_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
25	0x0	GPIO_17_ROW_EN: Indicates whether GPIO pin #17 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #17 in column configuration 0 = NOT_MAPPED 1 = MAPPED
24:21	0x0	GPIO_16_ROW_NUM: Mapping of GPIO pin #16 to row number. Valid only if GPIO_16_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
20	0x0	GPIO_16_ROW_EN: Indicates whether GPIO pin #16 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #16 in column configuration 0 = NOT_MAPPED 1 = MAPPED
19:16	0x0	GPIO_15_ROW_NUM: Mapping of GPIO pin #15 to row number. Valid only if GPIO_15_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
15	0x0	GPIO_15_ROW_EN: Indicates whether GPIO pin #15 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #15 in column configuration 0 = NOT_MAPPED 1 = MAPPED
14:11	0x0	GPIO_14_ROW_NUM: Mapping of GPIO pin #14 to row number. Valid only if GPIO_14_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
10	0x0	GPIO_14_ROW_EN: Indicates whether GPIO pin #14 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #14 in column configuration 0 = NOT_MAPPED 1 = MAPPED
9:6	0x0	GPIO_13_ROW_NUM: Mapping of GPIO pin #13 to row number. Valid only if GPIO_13_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
5	0x0	GPIO_13_ROW_EN: Indicates whether GPIO pin #13 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #13 in column configuration 0 = NOT_MAPPED 1 = MAPPED
4:1	0x0	GPIO_12_ROW_NUM: Mapping of GPIO pin #12 to row number. Valid only if GPIO_12_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
0	0x0	GPIO_12_ROW_EN: Indicates whether GPIO pin #12 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #12 in column configuration 0 = NOT_MAPPED 1 = MAPPED

17.5.6 APBDEV_KBC_ROW_CFG3_0

Fourth row configuration register. This register setting will take precedence in case the same GPIO pin is configured in the column register also. It does the mapping between the GPIO pins and rows of the keypad matrix. It configures for GPIO pin numbers 18 to 23.

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:1	0x0	GPIO_18_ROW_NUM: Mapping of GPIO pin #18 to row number. Valid only if GPIO_18_ROW_EN is set. Indicates row number. 0x0 = Row number 0. 0xF = Row number 15
0	0x0	GPIO_18_ROW_EN: Indicates whether GPIO pin #18 is mapped to any row of the keypad matrix. This bit overrides any setting done for pin #18 in column configuration 0 = NOT_MAPPED 1 = MAPPED

17.5.7 APBDEV_KBC_COL_CFG0_0

First column configuration register. It does the mapping between the GPIO pins and columns of the keypad matrix. It configures for GPIO pin numbers 0 to 7.

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:29	0x0	GPIO_7_COL_NUM: Mapping of GPIO pin #7 to column number. Valid only if GPIO_7_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
28	0x0	GPIO_7_COL_EN: Indicates whether GPIO pin #7 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_7_ROW_EN in ROW_CFG1 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
27:25	0x0	GPIO_6_COL_NUM: Mapping of GPIO pin #6 to column number. Valid only if GPIO_6_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
24	0x0	GPIO_6_COL_EN: Indicates whether GPIO pin #6 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_6_ROW_EN in ROW_CFG1 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
23:21	0x0	GPIO_5_COL_NUM: Mapping of GPIO pin #5 to column number. Valid only if GPIO_5_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
20	0x0	GPIO_5_COL_EN: Indicates whether GPIO pin #5 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_5_ROW_EN in ROW_CFG0 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
19:17	0x0	GPIO_4_COL_NUM: Mapping of GPIO pin #4 to column number. Valid only if GPIO_4_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
16	0x0	GPIO_4_COL_EN: Indicates whether GPIO pin #4 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_4_ROW_EN in ROW_CFG0 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
15:13	0x0	GPIO_3_COL_NUM: Mapping of GPIO pin #3 to column number. Valid only if GPIO_3_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
12	0x0	GPIO_3_COL_EN: Indicates whether GPIO pin #3 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_3_ROW_EN in ROW_CFG0 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
11:9	0x0	GPIO_2_COL_NUM: Mapping of GPIO pin #2 to column number. Valid only if GPIO_2_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7

Bit	Reset	Description
8	0x0	GPIO_2_COL_EN: Indicates whether GPIO pin #2 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_2_ROW_EN in ROW_CFG0 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
7:5	0x0	GPIO_1_COL_NUM: Mapping of GPIO pin #1 to column number. Valid only if GPIO_1_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
4	0x0	GPIO_1_COL_EN: Indicates whether GPIO pin #1 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_1_ROW_EN in ROW_CFG0 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
3:1	0x0	GPIO_0_COL_NUM: Mapping of GPIO pin #0 to column number. Valid only if GPIO_0_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
0	0x0	GPIO_0_COL_EN: Indicates whether GPIO pin #0 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_0_ROW_EN in ROW_CFG0 is set to 0. 0 = NOT_MAPPED 1 = MAPPED

17.5.8 APBDEV_KBC_COL_CFG1_0

Second column configuration register. It does the mapping between the GPIO pins and columns of the keypad matrix. It configures for GPIO pin numbers 8 to 15.

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:29	0x0	GPIO_15_COL_NUM: Mapping of GPIO pin #15 to column number. Valid only if GPIO_15_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
28	0x0	GPIO_15_COL_EN: Indicates whether GPIO pin #15 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_15_ROW_EN in ROW_CFG2 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
27:25	0x0	GPIO_14_COL_NUM: Mapping of GPIO pin #14 to column number. Valid only if GPIO_14_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
24	0x0	GPIO_14_COL_EN: Indicates whether GPIO pin #14 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_14_ROW_EN in ROW_CFG2 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
23:21	0x0	GPIO_13_COL_NUM: Mapping of GPIO pin #13 to column number. Valid only if GPIO_13_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
20	0x0	GPIO_13_COL_EN: Indicates whether GPIO pin #13 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_13_ROW_EN in ROW_CFG2 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
19:17	0x0	GPIO_12_COL_NUM: Mapping of GPIO pin #12 to column number. Valid only if GPIO_12_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
16	0x0	GPIO_12_COL_EN: Indicates whether GPIO pin #12 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_12_ROW_EN in ROW_CFG2 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
15:13	0x0	GPIO_11_COL_NUM: Mapping of GPIO pin #11 to column number. Valid only if GPIO_11_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7

Bit	Reset	Description
12	0x0	GPIO_11_COL_EN: Indicates whether GPIO pin #11 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_11_ROW_EN in ROW_CFG1 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
11:9	0x0	GPIO_10_COL_NUM: Mapping of GPIO pin #10 to column number. Valid only if GPIO_10_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
8	0x0	GPIO_10_COL_EN: Indicates whether GPIO pin #10 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_10_ROW_EN in ROW_CFG1 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
7:5	0x0	GPIO_9_COL_NUM: Mapping of GPIO pin #9 to column number. Valid only if GPIO_9_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
4	0x0	GPIO_9_COL_EN: Indicates whether GPIO pin #9 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_9_ROW_EN in ROW_CFG1 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
3:1	0x0	GPIO_8_COL_NUM: Mapping of GPIO pin #8 to column number. Valid only if GPIO_8_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
0	0x0	GPIO_8_COL_EN: Indicates whether GPIO pin #8 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_8_ROW_EN in ROW_CFG1 is set to 0. 0 = NOT_MAPPED 1 = MAPPED

17.5.9 APBDEV_KBC_COL_CFG2_0

Third column configuration register. It does the mapping between the GPIO pins and columns of the keypad matrix. It configures for GPIO pin numbers 16 to 23.

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:9	0x0	GPIO_18_COL_NUM: Mapping of GPIO pin #18 to column number. Valid only if GPIO_18_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
8	0x0	GPIO_18_COL_EN: Indicates whether GPIO pin #18 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_18_ROW_EN in ROW_CFG3 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
7:5	0x0	GPIO_17_COL_NUM: Mapping of GPIO pin #17 to column number. Valid only if GPIO_17_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
4	0x0	GPIO_17_COL_EN: Indicates whether GPIO pin #17 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_17_ROW_EN in ROW_CFG2 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
3:1	0x0	GPIO_16_COL_NUM: Mapping of GPIO pin #16 to column number. Valid only if GPIO_16_COL_EN is set. Indicates row number. 0x0 = Column number 0. 0x7 = Column number 7
0	0x0	GPIO_16_COL_EN: Indicates whether GPIO pin #16 is mapped to any column of the keypad matrix. This bit should be set to '1' only when GPIO_16_ROW_EN in ROW_CFG2 is set to 0. 0 = NOT_MAPPED 1 = MAPPED

17.5.10 APBDEV_KBC_TO_CNT_0

Continuous polling (CP) state time-out count register. This register holds the time-out count value.

In CP state, if the corresponding counter crosses this value KBC goes back to WuKP (Wake-up on Key-press state).

Offset: 0x24 | Read/Write: R/W | Reset: 0x00027100 (0bxxxxxxxxxxxxx00100111000100000000)

Bit	Reset	Description
19:0	0x27100	TO_CNT_VAL: Time-out count value. The default value is 5 seconds. The value should be calculated for a 32 KHz clock.

17.5.11 APBDEV_KBC_INIT_DLY_0

This register holds the initial delay for scan when KBC goes from WuKP mode to CP mode.

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000400 (0bxxxxxxxxxxxxx00000000010000000000)

Bit	Reset	Description
19:0	0x400	INIT_DLY_VAL: Initial delay value. The default value is 32.25 milliseconds. The value should be calculated for a 32 KHz clock.

17.5.12 APBDEV_KBC_RPT_DLY_0

This register holds the repeat scan delay. This is the delay between two successive scans in CP mode.

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000400 (0bxxxxxxxxxxxxx00000000010000000000)

Bit	Reset	Description
19:0	0x400	RPT_DLY_VAL: delay value. The default value is 32.25 milliseconds. The value should be calculated for a 32 KHz clock.

17.5.13 APBDEV_KBC_KP_ENT0_0

First key-press entries register. This register holds the key numbers of the keys which have been pressed. It holds the first four entries.

Note: Both entry registers should be read consecutively. A read of the first register changes the FIFO read pointer

Offset: 0x30 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	KP_NEW_ENT3: Indicates whether fourth entry is valid or not. 0x0 = Entry not valid. 0x1 = Valid entry
30:27	X	KP_ROW_NUM_ENT3: Row number for fourth key. 0x0 = Row number 0. 0xF = Row number 15
26:24	X	KP_COL_NUM_ENT3: Column number for fourth key. 0x0 = Column number 0. 0x7 = Column number 7
23	X	KP_NEW_ENT2: Indicates whether third entry is valid or not. 0x0 = Entry not valid. 0x1 = Valid entry
22:19	X	KP_ROW_NUM_ENT2: Row number for third key. 0x0 = Row number 0. 0xF = Row number 15
18:16	X	KP_COL_NUM_ENT2: Column number for third key. 0x0 = Column number 0. 0x7 = Column number 7
15	X	KP_NEW_ENT1: Indicates whether second entry is valid or not. 0x0 = Entry not valid. 0x1 = Valid entry

Bit	Reset	Description
14:11	X	KP_ROW_NUM_ENT1: Row number for second key. 0x0 = Row number 0. 0xF = Row number 15
10:8	X	KP_COL_NUM_ENT1: Column number for second key. 0x0 = Column number 0. 0x7 = Column number 7
7	X	KP_NEW_ENT0: Indicates whether first entry is valid or not. 0x0 = Entry not valid. 0x1 = Valid entry
6:3	X	KP_ROW_NUM_ENT0: Row number for first key. 0x0 = Row number 0. 0xF = Row number 15
2:0	X	KP_COL_NUM_ENT0: Column number for first key. 0x0 = Column number 0. 0x7 = Column number 7

17.5.14 APBDEV_KBC_KP_ENT1_0

Second key-press entries register. This register holds the key numbers of the keys which have been pressed. It holds the last four entries (out of eight entries)

Note: Both entry registers should be read consecutively. A read of the first register changes the FIFO read pointer

Offset: 0x34 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	KP_NEW_ENT7: Indicates whether eighth entry is valid or not. 0x0 = Entry not valid. 0x1 = Valid entry
30:27	X	KP_ROW_NUM_ENT7: Row number for eighth key. 0x0 = Row number 0. 0xF = Row number 15
26:24	X	KP_COL_NUM_ENT7: Column number for eighth key. 0x0 = Column number 0. 0x7 = Column number 7
23	X	KP_NEW_ENT6: Indicates whether seventh entry is valid or not. 0x0 = Entry not valid. 0x1 = Valid entry
22:19	X	KP_ROW_NUM_ENT6: Row number for seventh key. 0x0 = Row number 0. 0xF = Row number 15
18:16	X	KP_COL_NUM_ENT6: Column number for seventh key. 0x0 = Column number 0. 0x7 = Column number 7
15	X	KP_NEW_ENT5: Indicates whether sixth entry is valid or not. 0x0 = Entry not valid. 0x1 = Valid entry
14:11	X	KP_ROW_NUM_ENT5: Row number for sixth key. 0x0 = Row number 0. 0xF = Row number 15
10:8	X	KP_COL_NUM_ENT5: Column number for sixth key. 0x0 = Column number 0. 0x7 = Column number 7
7	X	KP_NEW_ENT4: Indicates whether fifth entry is valid or not. 0x0 = Entry not valid. 0x1 = Valid entry
6:3	X	KP_ROW_NUM_ENT4: Row number for fifth key. 0x0 = Row number 0. 0xF = Row number 15
2:0	X	KP_COL_NUM_ENT4: Column number for fifth key. 0x0 = Column number 0. 0x7 = Column number 7

17.5.15 APBDEV_KBC_ROW0_MASK_0

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	KBC_ROW0_COL7_MASK_ENABLE: Disable row0 col7 when the system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair. 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW0_COL6_MASK_ENABLE: Disable row0 col6 when the system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW0_COL5_MASK_ENABLE: Disable row0 col5 when the system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair. 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW0_COL4_MASK_ENABLE: Disable row0 col4 when the system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair. 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW0_COL3_MASK_ENABLE: Disable row0 col3 when the system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair. 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW0_COL2_MASK_ENABLE: Disable row0 col2 when the system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair. 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW0_COL1_MASK_ENABLE: Disable row0 col1 when the system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair. 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW0_COL0_MASK_ENABLE: Disable row0 col0 when the system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair. 0 = ENABLE 1 = DISABLE

17.5.16 APBDEV_KBC_ROW1_MASK_0

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	KBC_ROW1_COL7_MASK_ENABLE: Disable row1 col7 when system is in suspend/deep sleep mode. 1: Dmode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW1_COL6_MASK_ENABLE: Disable row1 col6 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW1_COL5_MASK_ENABLE: Disable row1 col5 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW1_COL4_MASK_ENABLE: Disable row1 col4 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
3	0x0	KBC_ROW1_COL3_MASK_ENABLE: Disable row1 col3 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW1_COL2_MASK_ENABLE: Disable row1 col2 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW1_COL1_MASK_ENABLE: Disable row1 col1 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW1_COL0_MASK_ENABLE: Disable row1 col0 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

17.5.17 APBDEV_KBC_ROW2_MASK_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	KBC_ROW2_COL7_MASK_ENABLE: Disable row2 col7 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW2_COL6_MASK_ENABLE: Disable row2 col6 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW2_COL5_MASK_ENABLE: Disable row2 col5 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW2_COL4_MASK_ENABLE: Disable row2 col4 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW2_COL3_MASK_ENABLE: Disable row2 col3 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW2_COL2_MASK_ENABLE: Disable row2 col2 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW2_COL1_MASK_ENABLE: Disable row2 col1 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW2_COL0_MASK_ENABLE: Disable row2 col0 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

17.5.18 APBDEV_KBC_ROW3_MASK_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	KBC_ROW3_COL7_MASK_ENABLE: Disable row3 col7 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW3_COL6_MASK_ENABLE: Disable row3 col6 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW3_COL5_MASK_ENABLE: Disable row3 col5 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW3_COL4_MASK_ENABLE: Disable row3 col4 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW3_COL3_MASK_ENABLE: Disable row3 col3 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW3_COL2_MASK_ENABLE: Disable row3 col2 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW3_COL1_MASK_ENABLE: Disable row3 col1 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW3_COL0_MASK_ENABLE: Disable row3 col0 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

17.5.19 APBDEV_KBC_ROW4_MASK_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	KBC_ROW4_COL7_MASK_ENABLE: Disable row4 col7 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW4_COL6_MASK_ENABLE: Disable row4 col6 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW4_COL5_MASK_ENABLE: Disable row4 col5 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW4_COL4_MASK_ENABLE: Disable row4 col4 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
3	0x0	KBC_ROW4_COL3_MASK_ENABLE: Disable row4 col3 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW4_COL2_MASK_ENABLE: Disable row4 col2 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW4_COL1_MASK_ENABLE: Disable row4 col1 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW4_COL0_MASK_ENABLE: Disable row4 col0 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

17.5.20 APBDEV_KBC_ROW5_MASK_0

Offset: 0x4c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	KBC_ROW5_COL7_MASK_ENABLE: Disable row5 col7 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW5_COL6_MASK_ENABLE: Disable row5 col6 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW5_COL5_MASK_ENABLE: Disable row5 col5 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW5_COL4_MASK_ENABLE: Disable row5 col4 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW5_COL3_MASK_ENABLE: Disable row5 col3 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW5_COL2_MASK_ENABLE: Disable row5 col2 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW5_COL1_MASK_ENABLE: Disable row5 col1 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW5_COL0_MASK_ENABLE: Disable row5 col0 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

17.5.21 APBDEV_KBC_ROW6_MASK_0

Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	KBC_ROW6_COL7_MASK_ENABLE: Disable row6 col7 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW6_COL6_MASK_ENABLE: Disable row6 col6 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW6_COL5_MASK_ENABLE: Disable row6 col5 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW6_COL4_MASK_ENABLE: Disable row6 col4 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW6_COL3_MASK_ENABLE: Disable row6 col3 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW6_COL2_MASK_ENABLE: Disable row6 col2 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW6_COL1_MASK_ENABLE: Disable row6 col1 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW6_COL0_MASK_ENABLE: Disable row6 col0 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

17.5.22 APBDEV_KBC_ROW7_MASK_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	KBC_ROW7_COL7_MASK_ENABLE: Disable row7 col7 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW7_COL6_MASK_ENABLE: Disable row7 col6 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW7_COL5_MASK_ENABLE: Disable row7 col5 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW7_COL4_MASK_ENABLE: Disable row7 col4 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
3	0x0	KBC_ROW7_COL3_MASK_ENABLE: Disable row7 col3 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW7_COL2_MASK_ENABLE: Disable row7 col2 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW7_COL1_MASK_ENABLE: Disable row7 col1 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW7_COL0_MASK_ENABLE: Disable row7 col0 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

17.5.23 APBDEV_KBC_ROW8_MASK_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	KBC_ROW8_COL7_MASK_ENABLE: Disable row8 col7 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW8_COL6_MASK_ENABLE: Disable row8 col6 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW8_COL5_MASK_ENABLE: Disable row8 col5 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW8_COL4_MASK_ENABLE: Disable row8 col4 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW8_COL3_MASK_ENABLE: Disable row8 col3 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW8_COL2_MASK_ENABLE: Disable row8 col2 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW8_COL1_MASK_ENABLE: Disable row8 col1 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW8_COL0_MASK_ENABLE: Disable row8 col0 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

17.5.24 APBDEV_KBC_ROW9_MASK_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	KBC_ROW9_COL7_MASK_ENABLE: Disable row9 col7 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW9_COL6_MASK_ENABLE: Disable row9 col6 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW9_COL5_MASK_ENABLE: Disable row9 col5 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW9_COL4_MASK_ENABLE: Disable row9 col4 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW9_COL3_MASK_ENABLE: Disable row9 col3 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW9_COL2_MASK_ENABLE: Disable row9 col2 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW9_COL1_MASK_ENABLE: Disable row9 col1 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW9_COL0_MASK_ENABLE: Disable row9 col0 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

17.5.25 APBDEV_KBC_ROW10_MASK_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	KBC_ROW10_COL7_MASK_ENABLE: Disable row10 col7 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW10_COL6_MASK_ENABLE: Disable row10 col6 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW10_COL5_MASK_ENABLE: Disable row10 col5 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW10_COL4_MASK_ENABLE: Disable row10 col4 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
3	0x0	KBC_ROW10_COL3_MASK_ENABLE: Disable row10 col3 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW10_COL2_MASK_ENABLE: Disable row10 col2 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW10_COL1_MASK_ENABLE: Disable row10 col1 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW10_COL0_MASK_ENABLE: Disable row10 col0 when system is in suspend/deep sleep mode. 1: Disable row/column pair. 0: Enable row/column pair 0 = ENABLE 1 = DISABLE

18.0 CPU

The NVIDIA® Tegra® 4 series processor CPU complex contains quad ARM® Cortex™-A15 CPUs in a 4-PLUS-1 configuration with a fifth architecturally identical power-saving Cortex-A15 Companion Core.

18.1 Cortex-A15 CPU

Cortex-A15 is an advanced processor design with many features for high instruction throughput. It integrates the L2 cache controller into the CPU complex unlike Cortex-A9. All of the CPUs include the NEON Media Processing Engine. Further details of the Cortex-A15 itself are available from ARM.

These two documents are the key references on Cortex-A15, and both are available from ARM's website:

- Cortex-A15
  Revision: r2p1
  Technical Reference Manual

  Published by ARM Limited, document number ARM DDI 0438D.
- ARM Architecture Reference Manual
  ARM v7-A and ARM v7-R edition

  Published by ARM Limited, document number ARM DDI 0406C.

Note: These are the current versions at the time of writing. You should periodically look for updates and always refer to the latest available version of this material.

You should also review the latest version of the ARM errata for Cortex-A15. We expect ARM to publish this on their website, but at the time of writing it is not available publicly. Consult your NVIDIA AE for more details.

18.1.1 Cortex-A15 Implementation Details

Many aspects of the Cortex-A15 may be configured for a specific implementation. The Tegra 4 implementation has the following features:

Cortex-A15 Processor Configuration		
	Option	
Feature	**Main Quad-Core Complex**	**Companion Core**
Processor Revision	r2p3	r2p3
Number of processors	4	1
L2 Cache Size	2MB	512KB
L2 Arbitration register slice	1	0
L2 Tag RAM register slice	1	1
L2 Data RAM register slice	1	1
L2 Tag RAM latency	2	2
L2 Data RAM latency	3 cycle	2 cycle
L2 Logic idle gated clock	Included	Included
ECC/parity support	None	None

Cortex-A15 Processor Configuration		
	Option	
Feature	Main Quad-Core Complex	Companion Core
VFP and Neon	Included	Included
Generic Interrupt Controller	Included	Included
Shared Peripheral Interrupts	160	160

18.2 4-Plus-1 Configuration and Control

The NVIDIA 4-PLUS-1 processor architecture has a fifth Companion Core available for low power operation. In addition to the main quad core CPU complex, the Companion Core is an additional CPU complex replicating CPU 0 of the main complex, and implemented in a low-leakage technology. The Companion Core is also sometimes referred to by its engineering name, the "shadow" core; and also is commonly referred to as the LP Core or LP CPU Cluster referring to its Low Power operating capability.

The goal of this architecture is to provide efficient CPU operation across a very broad range of processor loads, with the quad-core complex optimized for high-speed operation with low dynamic power, and the Companion Core optimized for low-speed operation where transistor leakage becomes an important factor in power. These two CPU complexes operate exclusively to each other, meaning the hardware does not support simultaneous operation in any form. The main quad core complex has 2 Megabytes of L2 cache RAM, the companion core as a separate 512 Kilobyte L2 cache RAM.

The quad core complex implements separate power-gating for each of the four cores, and also a fifth power gate domain for the shared logic and L2.

Refer to the Flow Controller section of this document for details on how this is controlled.

19.0 FLOW CONTROLLER

The flow controller provides the sequencing of hardware-controlled CPU power states for the main CPU complex and the AVP coprocessor (also known as COP). It also provides hardware support for CPU cluster switching between the fast quad-core complex and the companion core.

The flow controller receives CPU power-state requests from CPUs, prioritizes them, and sends power on/off requests to PMC, which powers gates/un-gates corresponding CPUs. It monitors per CPU interrupts and events to determine CPU wake condition and initiates the CPU wake based on wake events/interrupts.

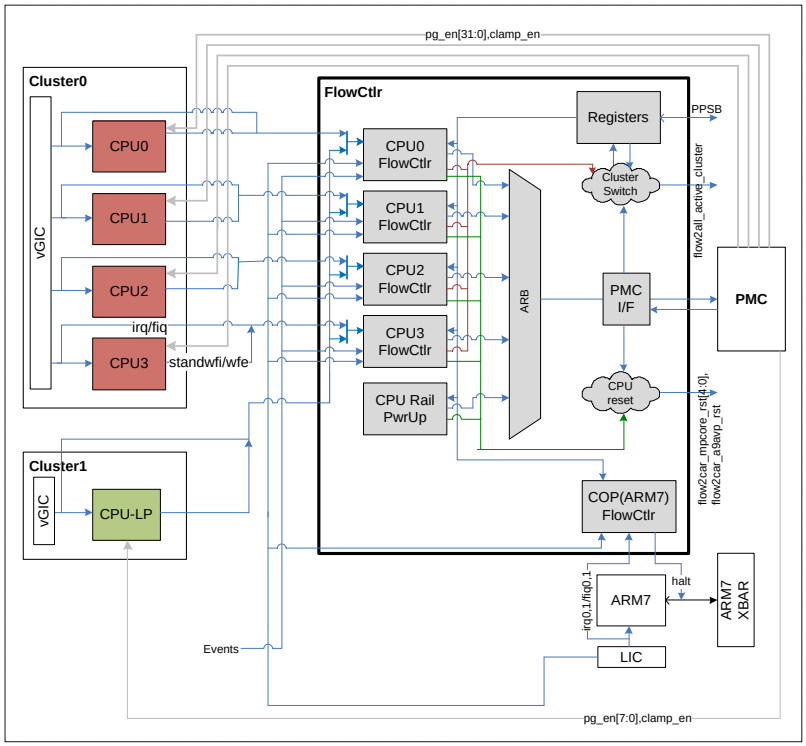
The flow controller provides configuration registers for software to configure wake events, etc. The flow controller provides a mechanism to halt conditionally or unconditionally:

- Conditional halt/resume is based on a variety of events, including timers, external trigger, and internal clocks.
- COP (AVP) is halted by extending a bus cycle.
- CPU can be halted by software interaction: software issues the WFE instruction to stop the processor, the flow controller asserts the EVENT input of the CPU to restart it.

19.1 Features and Functionality

The flow controller receives requests from processors (Cortex™-A15 processors and COP) to change their power state, or system power state. It also receives the interrupts from the interrupt controllers (vGIC and LIC) and other events which it uses to wake them up. It also provides support for cluster-switching.

Figure 47: Flow Controller Block Diagram



19.1.1 WFI/WFE Instructions and STANDBYWFI/WFE Signals Usage

The WFI (Wait For Interrupt) and WFE (Wait For Event) are the instructions of ARMv7-A architecture (supported by both Cortex-A15 and Cortex-A9 processors), which are used by the CPU to interlock with the flow controller for the purpose of triggering low-power-state transitions.

When executing WFx (WFI/WFE), the processor waits for all prior instructions to complete before entering the idle (low-power) state. The WFx instruction ensures that all explicit memory accesses that are prior to the WFx instruction in program order have completed. Also, WFx ensures that all speculative memory accesses have completed.

After executing WFI, the processor asserts STANDBYWFx signal. Assertion of STANDBYWFx guarantees that the processor is in idle, and there is no outstanding memory access. The processor continues to assert STANDBYWFI until one of the interrupt (IRQ or FIQ) is asserted or processor is reset. Similarly, the processor continues to assert STANDBYWFE until an “event” is received by processor.

The flow controller can be configured to trigger (per CPU) power-gating based on WFI instruction execution.

19.1.2 CPU (A15) Low-Power States

19.1.2.1 CPU Clock Gating

If a CPU is idle for a short period of time, then it can be halted which results in its CPU clock-gating. This is referred to as LP3 CPU power-state. The WFI instruction is executed for putting CPU into LP3 (i.e. for clock-gating). After executing WFI, CPU disables most of the clocks in the CPU except the clock for the small wake up logic clock.

Exit from halt state occurs when the CPU detects an interrupt on IRQ or FIQ pins. Note that this state is per CPU, and that this clock-gating is internal to the Cortex-A15 processor, so it does not require any support from flow-controller.

19.1.2.2 CPU Power Gating

The flow-controller uses a (separate) state machine to sequence power-gating for each CPU. The main CPUs flow-controller state machines are different from that for the CPU Rail PwrUp, and COP. So it has three different variants of state machines: one for main CPUs (which is instantiated 4 times, one for each CPU), one for CPU Rail PwrUp and one for COP. If the active cluster is cluster1, then based on CPU-ID, corresponding state-machine is used for CPU-LP.

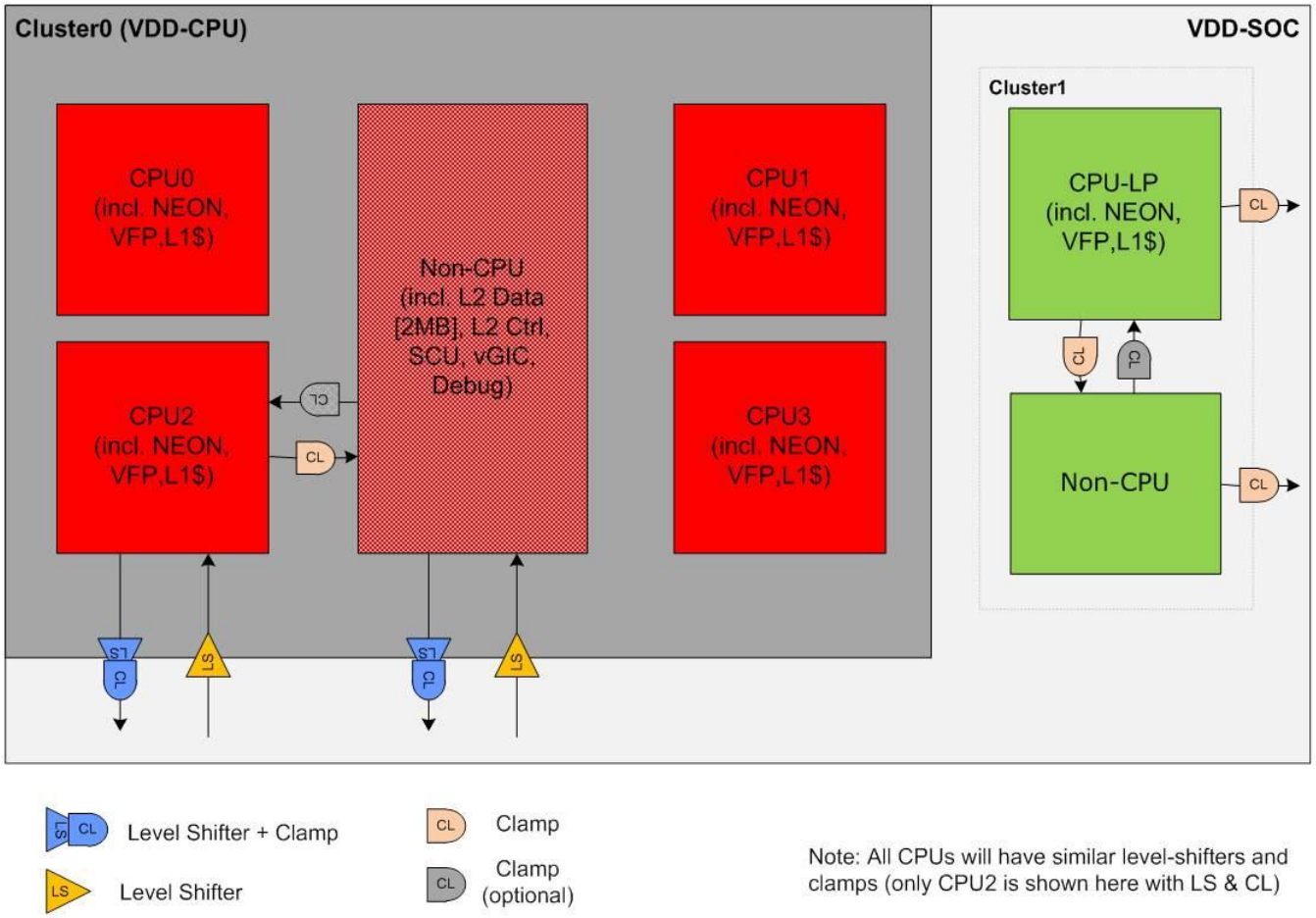
The flow controller can have one or more power-gating requests outstanding. However, it only issues one request at a time to PMC. A fixed priority arbiter is used to arbitrate among CPU power-gating on/off requests, and winning request is sent to PMC. This is primarily for two reasons: i) to avoid power noise issues, only one partition is power gated/ungated at a time, ii) PMC’s power-gating controller supports only one request at a time.

The flow controller to PMC interface includes core-ID which explicitly indicates which CPU or non-CPU partition should be power gated/ungated. PMC is not expected to use cluster-ID for identifying whether request is for cluster0 or cluster1 CPU.

CPU Power Gated Domains

As shown in the figure below, each CPU is an independent power-gated domain. The cluster0 non-CPU may be power gated as a separate domain. The cluster1 non-CPU is power-gated as a separate domain. The flow controller provides option for cluster0 L2 to be power gated.

Figure 48: CPUs PG Domains



19.1.2.3 Cluster0 CPU-Rail Gating

At boot, CPU rail is off by default, and is powered on by the AVP boot code using a **direct register write** to PMC (or by direct I2C commands to PMIC). Also, at LP0 exit, CPU rail (if need be) is powered-on (by SOFTWARE) by **direct register write** to PMC (or by direct I2C commands to PMIC).

The CPU rail is powered off (with the help of the flow controller) when last CPU is power-gated (see the section on “Cluster0 CPU Power Gating”). Similarly, when first CPU is woken up (by flow-controller based on wake-event) then CPU rail is first powered-on (if it was off) followed by CPU power-ungating (see the section on Cluster0 CPU Power Gating”).

19.1.2.4 Cluster0 CPU Power Gating

Each of the cluster0 CPUs can be independently power-gated with the help of the flow controller. If the CPU being power-gated is not the last one (or is not requesting the CPU rail to be powered off), then it is power gated/ungated as described in the section on “CPU Power Gating/Ungating Sequence”.

The last CPU being power-gated can request non-CPU to be power-gated or CPU rail to be powered off. Similar when the first CPU is being woken up, then the flow controller would power on (if it was off) CPU rail or power-ungate non-CPU (if it was power-gated).

Note:	The flow controller will only sequence CPU-rail or non-CPU (with last CPU power-gating request), but not both.

The flow controller can be set up to power gate/ungate each of the CPU independently. In addition, flow controller can be set up to power gate/ungate non-CPU (or power off/on CPU-rail) as result of CPU power gate/ungate request. The following section describes non-CPU power gate/ungate (or CPU power off/on) along with CPU. The individual CPU only power gate/ungate is described in the section on “CPU Power Gating/Ungating Sequence”.

Power-Gating (power-off) Last CPU and CPU Rail (or non-CPU)

Software initiates the last CPU power-gating request (with CPU-rail power-off or non-CPU power-gating). As a result, the flow-controller sequences CPU and CPU-rail (or non-CPU) power-gating as follows.

> **Note:** The following sequence is for last CPU power-gating with CPU rail power off option, but the same sequence will be used for CPU power-gating with non-CPU power-gating option.

- [SW-CPU] Initiate last CPU power gating with CPU rail power-off (section on CPU Power-Gating (Power OFF) Sequence has details of SOFTWARE steps)
- [HW-Flow] CPU power-gating step
  - Request PMC to power-gate the CPU
  - Wait for PMC ack
- [HW-Flow] CPU rail power-off step
  - If the CPU rail is off (see CPU_PWR_CSR[RAIL_STS]) or in the process of being powered on, then this step is done (skip the CPU rail power-off step).

> **Note:** For debug purposes, CPU_PWR_CSR[RAIL_STS] register bits can be updated to indicate CPU rail power off. This can help force the flow controller to skip rail power-off.
> The CPU rail may be in the process of being powered on because of another CPU wake up event. In that case, we don’t want this step to power-off CPU rail.

- Update CPU rail status register (CPU_PWR_CSR[RAIL_STS]) to be in the process powering off
- Request PMC to power off CPU rail (see section on “CPU Rail Gating/Ungating Sequence”)
- Waits for PMC acknowledge
- Update CPU rail power status (CPU_PWR_CSR[RAIL_STS]) to be powered off

The following table indicates possible power off options which can be triggered as a result of a CPU power-gating request.

Per CPU Pwr-Gate	Non-CPU Pwr-Gate	CPU Rail-Off	Comments
Yes	No	No	Individual CPU power-gating
Yes	Yes	No	
Yes	No	Yes	

Power-Ungating (power-on) first CPU and CPU Rail (and/or non-CPU)

Based on CPU wakeup-event (see HALT_CPUx_EVENT register), the flow controller power-ungates the CPU including the CPU rail or non-CPU power-off (if required).

- [HW-Flow] Receives CPU wakeup event
- [HW-Flow] CPU rail power-on step
  - If CPU rail is powered on, then this step is done. Else,
  - If CPU rail is in the process of being powered on, then wait for it to be powered on. After wait, this step is done. Else,
  - If CPU rail is in the process of being powered off, then wait for it to be powered off. After that continue to power it on.
  - Update CPU rail status register (CPU_PWR_CSR[RAIL_STS]) to be in the process of powering on.

- Request PMC to power on CPU rail
- Waits for PMC acknowledgment
- Update CPU rail status register (CPU_PWR_CSR[RAIL_STS]) to be powering on.

- [HW-Flow] Non-CPU power-ungating step
  - If non-CPU is power-ungated, then this step is done. Else,
  - If non-CPU is in the process of being powered on, then wait for it to be powered on. After wait, this step is done. Else,
  - If non-CPU is in the process of being powered off, then wait for it to be powered off. After that continue to power it on.
  - Update non-CPU pwr status register (CPU_PWR_CSR[C0NC_STS]) to be in the process of powering on.
  - Request PMC to power-ungate non-CPU
  - Wait for PMC acknowledgment
  - Update non-CPU pwr status register (CPU_PWR_CSR[RAIL_STS]) to be powered on.
  - Note, in terms of power-status, normally we would have either CPU rail or non-CPU to be off (but not both). However, SOFTWARE can program CPU-PWR_CSR register such that both the CPU rail and non-CPU will have to be powered on.
- [HW-Flow] CPU power-ungating step
  - Request PMC to power-ungate CPU
  - Waits for PMC acknowledgment

The following table indicates power-on options which can be triggered when the flow controller receives CPU wakeup event.

Per CPU Power Ungate	Non-CPU Power UnGate	CPU Rail Power On	Comments
Yes	No	No	Individual CPU power-ungating
Yes	Yes	No	
Yes	No	Yes	
Yes	Yes	Yes	

19.1.2.5 Cluster1 CPU Power Gating

The cluster1 CPU (also known as shadow CPU or CPU-LP) can be power gated/ungated with the help of the flow controller. Along with CPU-LP power-gating, its non-CPU can be power gated (by configuring flow controller before requesting CPU power-gating). From PMC perspective, CPU-LP and non-CPU are two separate power-gated domains. The flow controller sends CPU-LP and non-CPU power-gating requests to the PMC as follows:

The flow controller can be set up to power gate/ungate CPU-LP only. In addition, the flow controller can be set up to power gate/ungate non-CPU along with CPU-LP as result of CPU-LP power gate/ungate request. The next section describes non-CPU power gate/ungate along with CPU-LP. The CPU-LP only power gate/ungate is described in the section on “CPU Power Gating/Ungating Sequence”.

Power-gating (power-off) CPU and non-CPU

To initiate CPU-LP power-gating, SOFTWARE configures the flow-controller registers associated with CPU-ID (in MPID register). The SOFTWARE initiates CPU-LP (with non-CPU) power-gating request as described in section on CPU Power-Gating (Power OFF) Sequence. After that, flow-controller sequences CPU-LP and non-CPU power-gatings as follows:

- [SW-CPU] Initiate CPU-LP power gating with non-CPU power-gating rail (refer to the section on CPU Power-Gating (Power OFF) Sequence) for details of SOFTWARE steps

- [HW-Flow] CPU power-gating step
  - Request PMC to power-gate the CPU (Refer to section on “CPU Power-Gating (Power OFF) Sequence”)
  - Wait for PMC acknowledgment
- [HW-Flow] non-CPU power-gating step
  - If non-CPU is off (see CPU_PWR_CSR[C1NC_STS]), or in the process of being powered on or off then this step is done.

    Note: This condition should never be seen. But this makes the cluster1 sequence the same as that of cluster0, so it is all right to make this check in the state machine.
  - Update non-CPU pwr status register (CPU_PWR_CSR[C1NC_STS]) to be in the process powering off
  - Request PMC to power-gate non-CPU (refer to the section on “Non-CPU Power Gating/Ungating Sequence”)
  - Waits for PMC acknowledgment
  - Update non-CPU power status (CPU_PWR_CSR[C1NC_STS]) to be powered off

Power-Ungating (Power On) CPU and Non-CPU

Based on CPU-LP wake-up event ((see HALT_CPUx_EVENT register), the flow controller power-ungates CPU-LP (and non-CPU).

- [HW-Flow] Receives CPU-LP wake event
- [HW-Flow] Non-CPU power-ungating step
  - If non-CPU is power-ungated, then this step is done. Else,
  - If non-CPU is in the process of being powered on, then wait for it to be powered on. After that, this step is done. Else,
  - If non-CPU is in the process of being powered off, then wait for it to be powered off. After that continue to power it off.

    Note: The above two steps should never be encountered. But making these checks is all right and makes this sequence similar to corresponding cluster0 sequence.
  - Update non-CPU power status register (CPU_PWR_CSR[C0NC_STS]) to be in the process of powering on.
  - Request PMC to power-ungate non-CPU
  - Wait for PMC acknowledgment
  - Update non-CPU power status register (CPU_PWR_CSR[RAIL_STS]) to be powered on.
- [HW-Flow] CPU power-ungating step
  - Request PMC to power-ungate CPU
  - Waits for PMC ack

19.1.3 Cluster Switch

19.1.3.1 Cluster1 (Shadow) to Cluster0 Switch

The cluster switch is a variant of LP2 low-power state transition. From usage perspective, cluster1 to cluster0 switch is latency critical. For cluster1 to cluster0 switch, the cluser1 CPU is power-gated (with its non-CPU power-gating), cluster switched is performed, and a cluster0 CPU is powered on (either immediately or conditional on a wake event).

Following is the cluster1 to cluster0 migration sequence:

- [SW-CPU] Configure the flow controller for cluster switch
- [SW-CPU] Configure cluster0 CPU wake event
  - Software has to configure wake event for the CPU which is in the CPU-ID. After cluster switch, the flow controller can only wake-up the CPU which is pointed to by CPU-ID.
- [SW-CPU] Trigger **CPU rail on** sequence as described in “CPU Rail Gating/Ungating Sequence”.

  Note: This step allows the CPU rail (cluster0 power) to be powered up in parallel with the cluster1 power-gate sequence. However, cluster1 is still an active cluster (until the cluster state is switched by the flow controller).
- [SW-CPU] Initiate power-gating of cluster1 CPU (with non-CPU)
- [HW-Flow] Perform cluster1 non-CPU and CPU-LP power-gating as described in the section on “Cluster1 CPU Power Gating”
- [HW-Flow] After cluster1 power-gating, wait for pre-cluster-switch delay (programmed in a flow-controller register)
- [HW-Flow] Perform cluster switch
- [HW-Flow] Wait for post-cluster-switch-delay (programmed in a flow-controller register)
- [HW-Flow] Waits for the wake event (this could be immediate)
- [HW-Flow] Power-on first CPU of cluster0 as described in the section on Cluster0 CPU Power Gating

19.1.3.2 Cluster0 to Cluster1 (Shadow) Switch

The cluster switch is a variant of LP2 low-power state transition. For cluster0 to cluster1 switch, the last CPU of cluster0 is power-gated (with the option of powering off CPU rail), the cluster is switched, and cluster1 CPU is power ungated (either immediately or conditional on wake event).

Following is the cluster0 to cluster1 migration sequence:

- [SW-CPU] Configure the flow controller for cluster switch
- [SW-CPU] Configure cluster1 CPU wake event
- [SW-CPU] Configure the CPU-ID of CPU which is being migrated to CPU-LP
- [SW-CPU] Initiate power-gating of last CPU (with CPU rail off option) as described in section on “Cluster0 CPU Power Gating” for CPU power-gating and CPU rail gating.
- [HW-Flow] After CPU rail gating, wait for pre-cluster-switch delay (programmed in flow-controller register)
- [HW-Flow] Perform cluster switch
- [HW-Flow] Wait for post-cluster-switch-delay (programmed in flow-controller register)
- [HW-Flow] Waits for the wake event (if needed)
- At the wake event, flow controller powers up cluster1 as described in section on “Cluster1 CPU Power Gating”

19.1.4 CPU-ID Function

When the active cluster is cluster1, then the flow controller uses CPU-ID (programmed in MPID register) to associate CPU-LP to corresponding CPU state machine which is used to power gate/ungate CPU-LP. Also, during cluster1 to cluster0 switch, flow-controller uses CPU-ID to wake the corresponding cluster0 CPU (based on configured wake events).

Also, CPU-ID is used to drive cluster1 CPU-ID pins, which are captured in the CPU (CP15) register that software uses to read the CPU-ID of the current CPU. Note that the CP15 register captures the MPID value, which is the virtualized MPIDR, not the physical MPIDR. Thus, it must be read by the CPU in monitor mode or virtualization mode.

Once cluster0 is active and cluster migration has been completed, then the flow controller does not have any use of CPU-ID.

19.1.5 COP Low-Power States

19.1.5.1 Halt (Clock Gating)

There is no specific halt instruction in the COP. However, its memory bus cycle can be extended by any number of cycles.

When configured to put the COP into halt, the flow controller forces its memory bus interface into wait-state. After that if the COP issues any bus cycle (e.g., fetching next instruction), then that bus cycle does not return unless flow-controller releases its bus (i.e., wakes it up). The flow controller can be configured to wake up the COP by IRQ/FIQ or other wake up events. During a COP halt state, its clock is automatically gated by hardware.

19.1.5.2 Power-Gating

The COP is not power-gated.

19.1.5.3 CPU Power Gating/Ungating Sequence

The processor's (A15 or A9) power-gating can be triggered by either i) a direct register write to PMC or ii) interacting with the flow controller. This section describes the power-gating sequence via flow-controller interaction.

The flow controller interfaces with the Processors, PMC, and CAR to implement power-gating on/off hardware-sequence. The power-gating controller is in the PMC, while clock and reset controls are in CAR.

19.1.5.4 CPU Power-Gating (Power OFF) Sequence

- [SW-CPU] Prepare processor (flush CPU L1$, context save, set up Flow Controller) for power-gating
- [SW-CPU] Execute WFI to make sure there is no outstanding processor transaction
- [HW-CPU] Assert *STANDBYWFI* to the flow controller
- [HW-Flow] Assert flow2car_*_rst
  - This is not used in power gate sequence when powergate_enable is high
  - When powergate enable is "0", CAR uses this to assert CPU reset
- [HW-Flow] Request PMC (by asserting flow2pmc_req and associated bus) to power-off CPU
- [HW-PMC] Assert the clamp control signal. This goes to ccplex and also goes to CAR.
- [HW-CAR] Assert CPU reset and reset the CPU clock
  - [HW-CPU] Stop CPU clock
- [HW-CAR] Ack PMC to indicate CPU clock is stopped
- [HW-PMC] Power off CPU (assert CPU PG-enables)
- [HW-PMC] Clear PWRGATE_STS register bit. This is to indicate that CPU is power-gated.
- [HW-PMC] Ack to the flow controller to confirm power-off (by asserting pmc2flow_ack)

Note: After receiving ack from the PMC, the flow controller must not issue a new request to the PMC until pmc2flow_ack goes idle.

- [HW-Flow] Deassert flow2pmc_req (after seeing the Ack)
- [HW-PMC] Deassert Ack (after seeing request deassertion)
- [HW-Flow] Do not start another request until Ack goes down

19.1.5.5 CPU Power-Ungating (Power ON) Sequence

The generic CPU power-ungating (power OFF) sequence is described below. The specific details of main CPU (Cortex-A15) or COP1 (A9AVP) are described in their respective power-gating sections.

- [HW-Flow] After detecting wake up condition, request PMC to power-on CPU (by asserting flow2pmc_req and associated bus)
- [HW-PMC] Power-on the CPU (deassert CPU PG-enables)
- [HW-PMC] Set PWRGATE_STS register bit. This indicates that the CPU is power-ungated
- [HW-PMC] Ack flow controller (by asserting pmc2flow_ack) that CPU is powered on
- [HW-Flow] Deassert flow2car_*_rst to CAR
- [HW-CAR] Request CPU to ungate CPU clock (by deasserting car2ce*_cke)
- [HW-CPU] Ungate CPU clock
- [HW-CAR] Request PMC to remove CPU clamp (by asserting car2pmc_ack)
  - This is to make sure that clock gets ungated well ahead of reset deassertion
- [HW-PMC] Removes the clamping (by deasserting pmc2ce*_clamp)
  - This goes to the CPU and and also to the CAR
- [HW-PMC] Ack flow controller (by deasserting pmc2flow_ack)
- [HW-CAR] Deasserts reset
- [SW-CPU] CPU fetches code from reset vector.
- Note all of the Cortex-A15 CPUs share one reset-vector register (in EVP registers).

19.1.6 Non-CPU Power Gating/Ungating Sequence

From a software perspective, non-CPU power gating/ungating is done as part of CPU power gating/ungating request. The flow controller issues a separate request (to PMC) for power-gating CPU and non-CPU domain. The main reason is that PMC can only sequence one power domain at a time, so the flow controller serializes requests on behalf of PMC. From the flow controller hardware perspective, the sequence for non-CPU domain power gating is similar to CPU domain power gating/ungating which is described in above sections.

19.1.7 CPU Rail Gating/Ungating Sequence

This section only describes about CPU rail gating/ungating with the help of the flow controller.

From a software perspective, the CPU rail is powered on/off as part of CPU power-gating on/off request. The flow controller issues a separate request (to PMC) for CPU power gating/ungating and CPU rail gating/ungating. The main reason is that PMC can only sequence one power-domain at a time, so flow-controller serializes requests on behalf of PMC. From a flow-controller hardware perspective, the sequence for CPU rail gating is similar to individual CPU power gating, which is described in above sections. However, as part of CPU rail-ungating, flow controller also triggers RAM re-repair as described below.

19.1.7.1 CPU Rail Ungating Step

- [HW-Flow] Request the PMC to power-on the CPU rail
- [HW-Flow] Wait for PMC acknowledgment
- [HW-Flow] Request Reshift block (also known as re-repair block) to re-repair all of the segments of CPU repair chain
  - [HW-Reshift] Performs repair of all segments of CPU chain in parallel
- [HW-Flow] Wait for re-repair acknowledgments (from all segments)
- [HW-CAR] Deassert the CPU cluster0 reset

19.1.8 Cortex-A15 Debug Power Up/Down Requirements

The Cortex-A15 processor requires debug power up/down related support. The flow controller provides the following debug support.

19.1.8.1 DBGPWRUPREQ

The CoreSight can initiate power-up (for a CPU which is power-gated via the flow controller) by writing to the Cortex-A15 DBGPRCR register. As a result of this register write, A15 asserts DBGPWRUPREQ signal (a per CPU active-high level). Note, this signal can never be asserted when C0NC (or C1NC) is power-gated (since the logic generating this is inside those partitions).

The flow controller uses this signal as a wake-event (in addition to the wake-events programmed in CPUx_CSR register) to wake the corresponding CPU-only. The flow controller qualifies this signal by corresponding PWRUPREQ_QUAL bit (of FLOW_DBG_QUAL register). In summary, the flow controller uses the following condition to wake up a CPU.

*CEx wake-condition = (CEx wake-event as programmed in CPUx_CSR) || (<**ccplex2flow**>**_DBGPWRUPREQx && PWRUPREQ_QUALx**)*

19.1.8.2 DBGNOPWRDWN

The CoreSight can force CPU power-down to be power-down “emulation” only by writing to the Cortex-A15 DBGPRCR register. As a result of this register write, the Cortex-A15 asserts DBGNOPPWRDWN signal (a per CPU active-high level). Note, this signal is asserted to emulate CPU-only power-gating.

The flow controller qualifies <ccplex2flow>_DBGNOPPWRDWNx signal by corresponding NOPPWRDWN_QUALx bit (of FLOW_DBG_QUAL register) and uses that as an indication of CPU power-gating without the pg_enable option.

Note that CPU power-gating is still triggered by CPUx_CSR write only (based on WFI or WFE). But regardless of CPUx_CSR register configuration, if (<>_ DBGNOPPWRDWNx && NOPPWRDWN_QUALx == TRUE), then only CPUx is power-gated without the pg_enable option.

19.1.8.3 DBGPWRDWNREQ/DBGPWRDWNACK

Cortex-A15 debug logic needs to know about power-gating requests. It also needs to make sure that power-gating starts only after it has acknowledged (so it can flush trace buffers, etc.). For this purpose, the Cortex-A15 provides per CPU DBGPWRDWNREQ/ DBGPWRDWNACK interface.

Logically, STANDBYWFIx (Cortex-A15 output) can be fed to DBGPWRDWNREQx (Cortex-A15 input). But STANDBYWFI can be asserted for many reasons, so it has to be qualified when it is asserted for CPU power-gating. The flow controller provides qualifier (PWRDWNREQ_QUALx bits of FLOW_DBQ_QUAL register) which will be set by software at the power-gating entry, and cleared by software at CPU power-ungating.

Also, if qualifier (i.e., PWRDWNREQ_QUALx) is set, then the flow controller needs to wait for DBGPWRDWNACKx (in addition to waiting for existing STANDBYWFIx/WFEx condition) before starting the CPUx power-gating.

In summary, the following is needed in each ccplex cluster:

<ccplex>_ DBGPWRDWNREQx = <ccplex>_STANDBYWFIx & <flow2ccplex>_PWRDWNREQ_QUALx

The flow controller needs to wait for **<ccplex2flow>_DBGPWRDWNACKx & PWRDWNREQ_QUALx** (in addition to existing wait for STANDBYWFIx/STANDBYWFIx etc condition).

19.2 Flow Controller Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

In the Flow Controller register descriptions, CPU refers to CPU0, and COP refers to the AVP coprocessor. CPU-LP (CPU4) has its own set of control registers. This is required because there is a case (cluster1 to cluster0 switch) where both sets of registers can be configured in parallel. Also, since CPU-LP may morph into any of the four CPUs, it is better to keep its control register separate.

The EVENTS register is replicated per core, once for COP, four times for the cores of Cluster0.

The fields and enums have the following interpretation:

- MODE defines how the flow controller logic operates; the reset value is 0x0 for CPU0 and COP (does nothing) and 0x2 for CPU1, 2, 3 (WAITEVENT). This is related to the default behavior of the reset generator.
- Enumerated values for the field MODE are defined below.

Field Mode	Enumerated Value	Description
FLOW_MODE_NONE	0	No flow control
FLOW_MODE_RUN_AND_INT	1	Keep running but generate interrupt when event conditions met (not used)
FLOW_MODE_WAITEVENT	2	Stop running until event conditions met
FLOW_MODE_WAITEVENT_AND_INT	3	Same as FLOW_MODE_WAITEVENT but generate an interrupt when resumed (not used)
FLOW_MODE_STOP_UNTIL_IRQ	4	Stop until an interrupt controller interrupt occurs
FLOW_MODE_STOP_UNTIL_IRQ_AND_INT	5	Same as FLOW_MODE_STOP_UNTIL_INT but generate another interrupt when resumed (not used)
FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ	6	Stop until event conditions met AND an interrupt controller interrupt occurs

There is no enum for 7

- Bit 31: (also known as Halt) = This bit is set to configure the wake up (of corresponding CPU) by any of the interrupts (LIC_IRQ/FIQ, or GIC_RIQ/FIQ) corresponding to that CPU. If this bit is set, then none of the other fields of this register need to be set for CPU wake. This bit is set by software and cleared by hardware when any of these interrupt is received by hardware.
- Bit 30: (also known as Wait) = This bit is set to configure the wake up (of corresponding CPU) by any of the enabled "events". The events can be enabled by setting other fields of this register (incl. *_IRQ, *_FIQ). If this bit is set, then at least one of the events needs to be enabled to generate wake event. This bit is set by software and cleared by hardware when wait event is observed by hardware.
- Bit 29: (also known as Sync) = This bit causes the NEXT Flow Controller register read to stall the processor until the wake event occurs (Used for COP. Obsolete for CPUs).

The rest of the fields define which conditions will be taken into account to restart a halted processor.

There are two types of events:

- Counted events: Decrement the field called ZERO. Flow controller resumes when this counter reaches 0
- Activity events: The flow controller resumes when the activity occurs, no counting

All conditions are disabled at reset:

- JTAG: Resume on JTAG activity
- SCLK : Resume on Nth SYSCLK cycle ticks
- X32K : Resume on Nth X32K clock input ticks

- uSEC: Resume on Nth uSEC clock ticks (uSEC = microsecond)
- mSEC: Resume on Nth mSEC clock ticks (mSEC = millisecond)
- SEC : Resume on Nth second RTC clock ticks
- X_RDY: Resume on Nth XIO.RDY Ext. IO Ready events
- SMP3[1,0]: Resume on Nth SMP.3[1,0] Semaphore set events
- XRQ_[D,C,B,A] : Resume on Nth XRQ.[D,C,B,A] External Trigger events
- [O,I]B[E,F]: Resume on Nth [O,I]B[E,F] [Outbox, Inbox] [Empty, Full] Events
- [LIC, GIC]_[IRQ,FIQ]: Resume on [LIC, GIC] [IRQ,FIQ].
- ZERO: Initialized then decremented

Note: If more than one event is enabled, the event counter will decrement based on an OR condition of enabled events

19.2.1 FLOW_CTLR_HALT_CPU_EVENTS_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:29	0x0	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF

Bit	Reset	Description
13	0x0	IBE
12	0x0	IBF
11	0x0	LIC_IRQ
10	0x0	LIC_FIQ
9	0x0	GIC_IRQ
8	0x0	GIC_FIQ
7:0	0x0	ZERO

19.2.2 FLOW_CTLR_HALT_COP_EVENTS_0

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

Bit	Reset	Description
31:29	0x0	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE

Bit	Reset	Description
12	0x0	IBF
11	0x0	LIC_IRQ
10	0x0	LIC_FIQ
9	0x0	GIC_IRQ
8	0x0	GIC_FIQ
7:0	0x0	ZERO

19.2.3 FLOW_CTLR_CPU_CSR_0

The CPU_CSR registers are replicated for each CPU cores. They define additional characteristics of the flow controller linked to CPU cores, especially the interaction of the flow controller with power management functions. The LP1 state is normally entered and exited via side effects of the flow controller operation.

All fields of the CPU_CSR register have an initial value of 0. The different fields are:

- PWR_STATE : Current state of the PowerGate State Machine
- WAIT_EVENT: CPU is waiting, wake up is via an event
- HALT: CPU is halted
- P2F_ACK: pmc2flow_ack signal, this is the same signal for both CPU cores
- F2P_PWRUP: flow2pmc_pwrup, this is the same signal for both CPU cores
- F2P_REQ: flow2pmc_req valid, this is the same signal for both CPU cores
- F2C_MPCORE_RST: TRUE when requesting reset of MPCore™
- PWR_OFF_STS: TRUE when CPU PowerGated OFF by Flow Controller
- INTR_FLAG: TRUE when Interrupt is Active -- Write-1-to-Clear
- EVENT_FLAG: TRUE when Event is Active -- Write-1-to Clear
- ENABLE_EXT: If ENABLE is TRUE, then this specifies what to power off.
    - 00b: PowerGate CPU only.
    - 01b: PowerGate CPU and non-CPU (note, this option must be used only when last CPU is being power-gated)
    - 10b: PowerGate CPU and turn CPU rail off (note, this option is valid for cluster0 CPU, and must be used only when last CPU is being power-gated).
    - 11b: PG Emulation. Note, this option is for CPU-only PG emulation except for cluster-switch case (i.e. with SWITCH_CLUSTER=1) in which emulation applies to all PG steps of source cluster. In PG emulation, PG partition is reset but not power-gated.
- IMMEDIATE_WAKE: If set, CPU is powered up immediately (without waiting for an interrupt or event). Note: If this is set then HALT_CPUx_EVENTS[MODE] should be set to 0x2 for immediate wake.
- SWITCH_CLUSTER: If set, the active cluster will be switched when all indicated CPU reach STANDBY_WFI. This bit self clears once the cluster switch sequence is completed
- WAIT_WFI_BITMAP: All cores indicated in bitmap must be in STANDBY_WFI before switching Cluster
- WAIT_WFE_BITMAP: All cores indicated in bitmap must be in STANDBY_WFE before CPU PowerGating. This is deprecated and should not be used. It may work, but it is not being verified.
- EVENT_ENABLE: Generates an event when the flow controller exits the halted state
- ENABLE: PowerGate Enable - STANDBYWFx causes power-gating based on ENABLE_EXT values.

Note 1: Software sets this bit to '1' for requesting CPU power-gating. CPU is power-gated based on the value of ENABLE_EXT field.

Note 2: This bit is set by software and cleared by hardware (when power-gating sequence is completed).

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
27:24	RO	X	PWR_STATE
23	RO	X	WAIT_EVENT
22	RO	X	HALT
21	RO	X	P2F_ACK
20	RO	X	F2P_PWRUP
19	RO	X	F2P_REQ
17	RO	X	F2C_MPCORE_RST
16	RO	X	PWR_OFF_STS
15	RW	0x0	INTR_FLAG
14	RW	0x0	EVENT_FLAG
13:12	RW	0x0	ENABLE_EXT: 0 = POWERGATE_CPU_ONLY 1 = POWERGATE_BOTH_CPU_NONCPU 2 = POWERGATE_CPU_TURNOFF_CPURAIL 3 = PG_EMULATION
11:8	RW	0x0	WAIT_WFI_BITMAP
7:4	RW	0x0	WAIT_WFE_BITMAP
3	RW	0x0	IMMEDIATE_WAKE
2	RW	0x0	SWITCH_CLUSTER
1	RW	0x0	EVENT_ENABLE
0	RW	0x0	ENABLE

19.2.4 FLOW_CTLR_COP_CSR_0

COP Control/Status for Interrupts

R/W addr=6000:700c

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxxxxxxxxxx)

Bit	Reset	Description
15	0x0	INTR_FLAG: TRUE when Interrupt is Active -- Write-1-to-Clear

19.2.5 FLOW_CTLR_XRQ_EVENTS_0

XRQ Event Detect Selector Register

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	XRQ_D7_XRQ_D0: Setting a bit to 1 enables event triggering for the corresponding bit in GPIO port D. The assertion level is determined by GPIO_INT.LVL.D. If more than one XRQ.D bit is set, the events are ORed together. The resultant event is enabled by setting the XRQ.D bit in the HALT_CPU.EVENTS or HALT_COP.EVENTS registers.
23:16	0x0	XRQ_C7_XRQ_C0: Setting a bit to 1 enables event triggering for the corresponding bit in GPIO port C. The assertion level is determined by GPIO_INT.LVL.C. If more than one XRQ.C bit is set, the events are ORed together. The resultant event is enabled by setting the XRQ.C bit in the HALT_CPU.EVENTS or HALT_COP.EVENTS registers.
15:8	0x0	XRQ_B7_XRQ_B0: Setting a bit to 1 enables event triggering for the corresponding bit in GPIO port B. The assertion level is determined by GPIO_INT.LVL.B. If more than one XRQ.B bit is set, the events are ORed together. The resultant event is enabled by setting the XRQ.B bit in the HALT_CPU.EVENTS or HALT_COP.EVENTS registers.
7:0	0x0	XRQ_A7_XRQ_A0: Setting a bit to 1 enables event triggering for the corresponding bit in GPIO port A. The assertion level is determined by GPIO_INT.LVL.A. If more than one XRQ.A bit is set, the events are ORed together. The resultant event is enabled by setting the XRQ.A bit in the HALT_CPU.EVENTS or HALT_COP.EVENTS registers.

19.2.6 FLOW_CTLR_HALT_CPU1_EVENTS_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x40000000 (0b01000000000000000000000000000000)

Bit	Reset	Description
31:29	0x2	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D

Bit	Reset	Description
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0x0	LIC_IRQ
10	0x0	LIC_FIQ
9	0x0	GIC_IRQ
8	0x0	GIC_FIQ
7:0	0x0	ZERO

19.2.7 FLOW_CTLR_CPU1_CSR_0

Offset: 0x18 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
27:24	RO	X	PWR_STATE
23	RO	X	WAIT_EVENT
22	RO	X	HALT
21	RO	X	P2F_ACK
20	RO	X	F2P_PWRUP
19	RO	X	F2P_REQ
17	RO	X	F2C_MPCORE_RST
16	RO	X	PWR_OFF_STS
15	RW	0x0	INTR_FLAG
14	RW	0x0	EVENT_FLAG
13:12	RW	0x0	ENABLE_EXT: 0 = POWERGATE_CPU_ONLY 1 = POWERGATE_BOTH_CPU_NONCPU 2 = POWERGATE_CPU_TURNOFF_CPURAIL 3 = PG_EMULATION
11:8	RW	0x0	WAIT_WFI_BITMAP
7:4	RW	0x0	WAIT_WFE_BITMAP
3	RW	0x0	IMMEDIATE_WAKE
2	RW	0x0	SWITCH_CLUSTER
1	RW	0x0	EVENT_ENABLE

Bit	R/W	Reset	Description
0	RW	0x0	ENABLE

19.2.8 FLOW_CTLR_HALT_CPU2_EVENTS_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x40000000 (0b01000000000000000000000000000000)

Bit	Reset	Description
31:29	0x2	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0X0	LIC_IRQ
10	0X0	LIC_FIQ
9	0X0	GIC_IRQ
8	0X0	GIC_FIQ
7:0	0X0	ZERO

19.2.9 FLOW_CTLR_CPU2_CSR_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
27:24	RO	X	PWR_STATE
23	RO	X	WAIT_EVENT
22	RO	X	HALT
21	RO	X	P2F_ACK
20	RO	X	F2P_PWRUP
19	RO	X	F2P_REQ
17	RO	X	F2C_MPCORE_RST
16	RO	X	PWR_OFF_STS
15	RW	0x0	INTR_FLAG
14	RW	0x0	EVENT_FLAG
13:12	RW	0x0	ENABLE_EXT: 0 = POWERGATE_CPU_ONLY 1 = POWERGATE_BOTH_CPU_NONCPU 2 = POWERGATE_CPU_TURNOFF_CPURAIL 3 = PG_EMULATION
11:8	RW	0x0	WAIT_WFI_BITMAP
7:4	RW	0x0	WAIT_WFE_BITMAP
3	RW	0x0	IMMEDIATE_WAKE
2	RW	0x0	SWITCH_CLUSTER
1	RW	0x0	EVENT_ENABLE
0	RW	0x0	ENABLE

19.2.10 FLOW_CTLR_HALT_CPU3_EVENTS_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x40000000 (0b01000000000000000000000000000000)

Bit	Reset	Description
31:29	0x2	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK

Bit	Reset	Description
26	0x0	X32K
25	0x0	uSEC
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0X0	LIC_IRQ
10	0X0	LIC_FIQ
9	0X0	GIC_IRQ
8	0X0	GIC_FIQ
7:0	0X0	ZERO

19.2.11 FLOW_CTLR_CPU3_CSR_0

Offset: 0x28 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	R/W	Reset	Description
27:24	RO	X	PWR_STATE
23	RO	X	WAIT_EVENT
22	RO	X	HALT
21	RO	X	P2F_ACK
20	RO	X	F2P_PWRUP
19	RO	X	F2P_REQ
17	RO	X	F2C_MPCORE_RST
16	RO	X	PWR_OFF_STS
15	RW	0x0	INTR_FLAG
14	RW	0x0	EVENT_FLAG

Bit	R/W	Reset	Description
13:12	RW	0x0	ENABLE_EXT: 0 = POWERGATE_CPU_ONLY 1 = POWERGATE_BOTH_CPU_NONCPU 2 = POWERGATE_CPU_TURNOFF_CPURAIL 3 = PG_EMULATION
11:8	RW	0x0	WAIT_WFI_BITMAP
7:4	RW	0x0	WAIT_WFE_BITMAP
3	RW	0x0	IMMEDIATE_WAKE
2	RW	0x0	SWITCH_CLUSTER
1	RW	0x0	EVENT_ENABLE
0	RW	0x0	ENABLE

19.2.12 FLOW_CTLR_CLUSTER_CONTROL_0

Offset: 0x2c │ Read/Write: R/W │ Reset: 0x04004000 (0b00000100000000000001000000xxxxxxx0)

Bit	Reset	Description
31:20	0x40	POST_SWITCH_DELAY: Two delays on top of the built in pipeline delay when switching cluster, in clock cycles
19:8	0x40	PRE_SWITCH_DELAY
0	0x0	ACTIVE: The Active field can be written, this should only be done by the COP after making sure all clusters are inactive. Directly writing the Active field will otherwise most probably result in serious errors as there are no interlock in hardware as when the flow controller is used to control the switch together with the use of WFI on the currently inactive cluster 0 = G 1 = LP

19.2.13 FLOW_CTLR_HALT_COP1_EVENTS_0

Offset: 0x30 │ Read/Write: R/W │ Reset: 0x40000000 (0b01000000000000000000000000000000)

Bit	Reset	Description
31:29	0x2	MODE: 0 = FLOW_MODE_NONE 1 = FLOW_MODE_RUN_AND_INT 2 = FLOW_MODE_WAITEVENT 3 = FLOW_MODE_WAITEVENT_AND_INT 4 = FLOW_MODE_STOP_UNTIL_IRQ 5 = FLOW_MODE_STOP_UNTIL_IRQ_AND_INT 6 = FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ 2 = FLOW_MODE_STOP 3 = FLOW_MODE_STOP_AND_INT 4 = FLOW_MODE_STOP_UNTIL_INT 5 = FLOW_MODE_STOP_UNTIL_INT_AND_INT 6 = FLOW_MODE_STOP_OR_INT
28	0x0	JTAG
27	0x0	SCLK
26	0x0	X32K
25	0x0	uSEC

Bit	Reset	Description
24	0x0	MSEC
23	0x0	SEC
22	0x0	X_RDY
21	0x0	SMP31
20	0x0	SMP30
19	0x0	XRQ_D
18	0x0	XRQ_C
17	0x0	XRQ_B
16	0x0	XRQ_A
15	0x0	OBE
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0x0	LIC_IRQ
10	0x0	LIC_FIQ
9	0x0	GIC_IRQ
8	0x0	GIC_FIQ
7:0	0x0	ZERO

19.2.14 FLOW_CTLR_COP1_CSR_0

Offset: 0x34 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00xxxxx0xxx00x00)

Bit	R/W	Reset	Description
27:24	RO	X	PWR_STATE
23	RO	X	WAIT_EVENT
22	RO	X	HALT
21	RO	X	P2F_ACK
20	RO	X	F2P_PWRUP
19	RO	X	F2P_REQ
17	RO	X	F2C_MPCORE_RST
16	RO	X	PWR_OFF_STS
15	RW	0x0	INTR_FLAG
14	RW	0x0	EVENT_FLAG
8	RW	0x0	WAIT_WFI_BITMAP
4	RW	0x0	WAIT_WFE_BITMAP

Bit	R/W	Reset	Description
3	RW	0x0	IMMEDIATE_WAKE
1	RW	0x0	EVENT_ENABLE
0	RW	0x0	ENABLE

19.2.15 FLOW_CTLR_CPU_PWR_CSR_0

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000100 (0bxxxxxxxxxxxxxxxxxxxxxxxx100000000)

Bit	Reset	Description
8	0x1	CPU_RG_CFG: If this bit is set, then the flow controller would initiate last-CPU PG (along with CPU RG (rail-gating) or non-CPU PG) only when other 3 CPUs are already power-gated. This is primarily for debug purposes. Note that the flow controller uses PMC pwrgate-status of CPU partitions to identify whether CPUs are power-gated or not. 0 = DISABLE 1 = ENABLE
7:6	0x0	C1NC_STS: Hardware updates this register based on the current status of the C1NC partition. Before requesting a power-off (or on) request, flow controller updates this status to "in the process" status. Note that the flow controller requests C1NC power on/off based on this register value or PMC pwrgate-status of C1NC depending upon the USE_FLOW_STS register. For debug purposes, this can be written by software. 0 = PARTITION_OFF 1 = PG_IN_PROGRESS 2 = PU_IN_PROGRESS 3 = PARTITION_ON
5:4	0x0	C0NC_STS: hardware updates this register based on the current status of the C0NC partition. Before requesting a power-off (or on) request, flow controller updates this status to "in the process" status. Note, flow controller requests C0NC power on/off based on this register value or PMC pwrgate-status of C0NC depending upon USE_FLOW_STS register. For debug purposes, this can be written by software. 0 = PARTITION_OFF 1 = PG_IN_PROGRESS 2 = PU_IN_PROGRESS 3 = PARTITION_ON
3	0x0	USE_FLOW_STS:If this is set (=1), then the flow controller will use C0NC_STS/ C1NC_STS/RAIL_STS bits to determine whether to power-gate C0NC/C1NC/CRAIL domains. If this bit is clear (=0), then the flow controller will use the PMC pwrgate-status of C0NC/C1NC/CRAIL to determine whether to power-gate the C0NC (and C1NC) domains. In either case, the C0NC_STS/C1NC_STS/RAIL_STS fields are updated in the same way.
2:1	0x0	RAIL_STS: Hardware updates this register based on the current status of the CPU-rail. Before requesting a power-off (or on) request, flow controller updates this status to "in the process" status. Note, flow controller requests CPU-RAIL power on/off based on this register value or PMC pwrgate-status of CRAIL depending upon USE_FLOW_STS register. For debug purposes, this can be written by software. 0 = RAIL_OFF 1 = RG_IN_PROGRESS 2 = RU_IN_PROGRESS 3 = RAIL_ON
0	0x0	RAIL_ENABLE: CPU rail power-on request. Software sets this to request CPU rail pwr-on by flow-controller interaction. As a result of this request, CPU rail is powered up and RAM-re-repair is completed. If rail is already on, writing to this is ignored. Hardware clears this bit when CPU power rail in turned on (including RAM re-repair is done). Note: For cluster1 -> cluster0 switch, software would request CPU rail power-on, followed by cluster1 context save, followed by cluster-switch request.

19.2.16 FLOW_CTLR_MPID_0

Offset: 0x3c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	CPU_ID: CPU-ID of C1 CPU. Software programs this field before cluster0 -> cluster1 switch. This ID is provided to MPCore_LP which is what is read when OS reads MPIDR. Also, flow-controller uses this to identify wake interrupts to be used for CPULP wakeup.

19.2.17 FLOW_CTLR_RAM_REPAIR_0

Offset: 0x40 │ Read/Write: R/W │ Reset: 0x00000004 (0bxxxxxxxxxxxxxxxx00000000xxxx01x0)

Bit	R/W	Reset	Description
23:16	RO	X	DBG_STS: Repair done (acknowledge) from repair logic.
15:8	RW	0x0	DBG_REQ: Repair request for individual segments. This field is valid only when DBG_EN is set to '1'. Bit[0]: segment0 of repair chain, Bit[1]: segment1 of repair chain, ..., Bit[7]: segment7 of repair chain
3	RW	0x0	DBG_EN: Debug enable to be able to repair of individual segments of the cluster0 repair chain 0 = DISABLE 1 = ENABLE
2	RW	0x1	BYPASS_EN: RAM repair bypass enable. 0: Flow-controller requests RAM-repair at CPU RAIL power-up. 1: Flow-controller does not request RAM-repair at CPU RAIL power-up 0 = DISABLE 1 = ENABLE
1	RO	X	STS: Indicates Cluster repair chain status. hardware sets this to '1' when repair of all segments is done. hardware clears this to '0' when RAM repair is in progress (either due to hardware or software request)
0	RW	0x0	REQ: Cluster0 RAM repair request. Software sets this bit to '1' to initiate RAM repair request to all segments in parallel. Note, software will have to trigger RAM repair in all cases when software powers-on CPU-rail (via direct register write). If CPU-rail is powered on via the flow controller, then hardware (flow controller) takes care of RAM repair. hardware clears this bit when software triggered repair is done (i.e. STS=1) 0 = DISABLE 1 = ENABLE

19.2.18 FLOW_CTLR_FLOW_DBG_SEL_0

Offset: 0x44 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxx0000)

Bit	R/W	Reset	Description
19:18	RO	X	RG_PWR_STATE: Current state of Rail Gate state machine
17:16	RO	X	RU_PWR_STATE: Current state of Rail Ungate state machine
15:14	RO	X	NC_PG_PWR_STATE: Current state of Non CPU Power Gate state machine
13:12	RO	X	NC_PU_PWR_STATE: Current state of Non CPU Power Ungate state machine

Bit	R/W	Reset	Description
11:8	RW	0x0	CNT1_SEL:Activity selection which would be counted by CNT1. **Note** power/rail activities are counted only when power/rail gating is via flow-controller (and not direct PMC register write). 0: Idle (counter holds its current value) 1: Increment by 1 every time CPU0 is power gated 2: Increment by 1 every time CPU1 is power gated 3: Increment by 1 every time CPU2 is power gated 4: Increment by 1 every time CPU3 is power gated 5: Increment by 1 every time and of CPU0/1/2 or 3 is power-gated 6: Increment by 1 every time CPU-LP is power gated 7: Increment by 1 every time Ccluster0 non-CPU is power gated 8: Increment by 1 every time Cluster1 non-CPU is power gated 9: Increment by 1 every time CPU RAIL is powered off 10: Increment by 1 every time cluser0 is switched to cluster1 11: Increment by 1 every time cluser1 is switched to cluster0 0 = IDLE 1 = CPU0_PG 2 = CPU1_PG 3 = CPU2_PG 4 = CPU3_PG 5 = CPU0123_PG 6 = CPULP_PG 7 = C0NC_PG 8 = C1NC_PG 9 = CRAIL_OFF 10 = C0_TO_C1_SWITCH 11 = C1_TO_C0_SWITCH
3:0	RW	0x0	CNT0_SEL: Activity selection which would be counted by CNT0. **Note:** Power/rail activities are counted only when power/rail gating is via the flow controller (and not direct PMC register write). 0: Idle (counter holds its current value) 1: Increment by 1 every time CPU0 is power gated 2: Increment by 1 every time CPU1 is power gated 3: Increment by 1 every time CPU2 is power gated 4: Increment by 1 every time CPU3 is power gated 5: Increment by 1 every time and of CPU0/1/2 or 3 is power-gated 6: Increment by 1 every time CPU-LP is power gated 7: Increment by 1 every time Ccluster0 non-CPU is power gated 8: Increment by 1 every time Cluster1 non-CPU is power gated 9: Increment by 1 every time CPU RAIL is powered off 10: Increment by 1 every time cluser0 is switched to cluster1 11: Increment by 1 every time cluser1 is switched to cluster0 0 = IDLE 1 = CPU0_PG 2 = CPU1_PG 3 = CPU2_PG 4 = CPU3_PG 5 = CPU0123_PG 6 = CPULP_PG 7 = C0NC_PG 8 = C1NC_PG 9 = CRAIL_OFF 10 = C0_TO_C1_SWITCH 11 = C1_TO_C0_SWITCH

19.2.19 FLOW_CTLR_FLOW_DBG_CNT0_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE: Activity count value (updated by hardware based on CNT0_SEL). This register can be written by software to be able to clear it.

19.2.20 FLOW_CTLR_FLOW_DBG_CNT1_0

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

Bit	Reset	Description
31:0	0x0	VALUE: Activity count value (updated by hardware based on CNT1_SEL). This register can be written by software to be able to clear it.

19.2.21 FLOW_CTLR_FLOW_DBG_QUAL_0

Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx0000xxx00000xxx00000xxx00000)

Bit	Reset	Description
27:24	0x0	AXICIF_CG_DIS: CCPLEX AXICIF/MCCIF clock gating disable. Bit[0]: Fast cluster rclk-gating disable (1=DISABLE, 0=ENABLE). Bit[1]: Fast cluster wclk-gating disable (1=DISABLE, 0=ENABLE). Bit[2]: Slow cluster rclk-gating disable (1=DISABLE, 0=ENABLE). Bit[3]: Slow cluster wclk-gating disable (1=DISABLE, 0=ENABLE)
20:16	0x0	PWRUPREQ_QUAL: CPU DBGPWRUPREQ qualifier. Bit encoding similar to PWRDNREQ_QUAL.
12:8	0x0	NOPWRDWN_QUAL: CPU DBGNOPPWRDN qualifier. Bit encoding similar to PWRDNREQ_QUAL.
4:0	0x0	PWRDNREQ_QUAL: CPU DBGPWRDNREQ qualifier. Bit [0] for CPU0. Bit [1] for CPU1. Bit [2] for CPU2. Bit [3] for CPU3. Bit [4] for CPU-LP (also known as shadow CPU)

19.2.22 FLOW_CTLR_FLOW_CTLR_SPARE_0

Offset: 0x54 | Read/Write: R/W | Reset: 0xffff0000 (0b11111111111111110000000000000000)

Bit	Reset	Description
31:16	0xffff	SPARE_HI
15:0	0x0	SPARE_LO

20.0 MEMORY CONTROLLER

Tegra® 4 devices feature a dual-channel memory controller with two separate 32-bit DRAM interfaces. These are interleaved to provide high performance with the load shared across both channels.

The Tegra 4 memory controller (MC) handles memory requests from internal clients and arbitrates among them to allocate memory bandwidth for DDR3, LPDDR2, and LPDDR3 SDRAMs. There are two external memory controller (EMC) modules, which interface with the external DDR/LPDDR devices.

Key features in the Tegra 4 memory controller include:

- Enhanced arbiter design for higher memory efficiency
- System Memory Management Unit (SMMU) for virtual to physical address mapping for any device
- Support for DDR3, low-voltage DDR3, LPDDR2, and LPDDR3 SDRAMs
- Dual-channel support (2 x 32-bit interfaces)
- QUSE functionality for qualification of the tristatable DQS clock in SDRAMs
- CA training for LPDDR3 devices with byte/bit swizzling

The memory interface speed varies with memory type, and the specific Tegra 4 SKU, so is not stated in this document.

20.1 Memory Controller Architecture

The memory controller architecturally consists of the following parts:

- Arbitration Domains, which can handle a single request or response per clock from a group of clients. Typically, a system has a single Arbitration Domain, but an implementation may divide the client space into multiple Arbitration Domains to increase the effective system bandwidth.
- Protocol Arbiter, which manage a related pool of memory devices. A system may have a single Protocol Arbiter or multiple Protocol Arbiters.
- Memory Crossbar, which routes request and responses between Arbitration Domains and Protocol Arbiters. In the simplest version of the system, the Memory Crossbar is just a pass through between a single Arbitration Domain and a single Protocol Arbiter.
- Global Resources, which include things like configuration registers which are shared across the Memory Subsystem.

20.2 Hardware Features

20.2.1 DRAM Protocol Arbiter Features

- 1 channel, 32 bits of data bus, 16 MB to 2 GB attached DRAM
  - 2 chip selects, up to 2 GB attached DRAM per chip select
  - 2 individually controllable clock-enables
  - 2 individually controllable ODTs (for DDR3)
- Each chip select may support different geometries
  - 3 bank bits
  - DDR3: up to 15 bits of row address if using 2 chip selects, up to 16 bits if using 1 chip select
  - Support for up to 11 column bits[1], i.e., up to 4KB page-size

[1] This includes C0, which is not pinned out on the DRAM, and C1, which will be zero for all burst-length four bursts.

- Per-byte data masks
- BL4 or BL8 for both DDR3 and LPDDR2. BL8 for LPDDR3. For DDR3, BC4_on_the_fly.
- Up to 8 Gb DDR3 devices. 8 Gb x 16 and 4 Gb x 8 devices are restricted on single-channel configurations
- Up to 8 Gb LPDDR2/LPDDR3 devices, supported as single or double rank, single-channel or dual-channel configurations
- Support for low-power modes:
  - Software controllable entry/exit from: self-refresh, power down, deep power down
  - Hardware dynamic entry/exit from: power-down, self-refresh
  - Support for intermittent or disabled DLL
  - Disable unused address/command taps based on whether in POP or Discrete mode, and whether Discrete is in in DDR3 or LPDDR2 mode.
  - Pads use DPD-mode during idle periods
- For Tegra 4 devices, trims can be set/adjusted per-byte AND per-chip-select. The PVT-compensated value can still be used and a PVT/frequency-compensated adjustment can be added to it (1/16, 1/8, 3/16 of a cycle) or a non-PVT-offset can be added. There is no hardware training mechanism. Either software or characterization needs to determine the best settings.
- Support for x32, x16, or x8 chips attached to the channel
  - DQ/DQS swizzling: MRR_BYTESEL need to match byte swizzling.

20.2.2 Memory Crossbar

The Memory Crossbar (MXB) is in charge of adapting Arbitration Domains to Protocol Arbiters. In a simple system, the MXB is a direct connection. In a more complex system, the MXB does the following:

- Conflict arbitration for ingress of requests to a single PA from multiple ADs.
- Conflict arbitration for egress of responses to a single AD from multiple PAs, with skid buffering to handle backpressure.
- Clock-domain crossing FIFOs for requests and responses destined for PAs that run on an asynchronous clock.
- Fragmentation of wide requests from a wider AD into a narrower PA.
- Reassembly of wide responses from a narrower PA to a wider AD.

20.2.3 Global Resources

The Memory Subsystem also contains some global resources not owned by a particular AD or PA:

- SMMU
- Configuration registers and statistics counters

20.2.4 Supported SDRAM Topologies

The following table lists the supported topologies for DDR3 devices. The EMC shares CS1 with A15. It supports either single rank configurations with up to 16 row address bits or two rank configurations with up to 15 row bits.

Table 49: Supported DDR3 Topologies

DRAM Configuration				**DRAM Device Information**					
Density (GB)	Number of Channels	Number of Ranks Per Channel	Number of Devices Per Rank	Density (Gb)	Package	Page Size (Kb)	Bank Width	Row Width	Col Width

DRAM Configuration				DRAM Device Information					
1	2	1	2	2	x16	2	3	14	10
2	2	1	2	4	x16	2	3	15	10
2	2	1	4	2	x8	1	3	15	10
4	2	1	4	4	x8	1	3	16	10
2	2	2	2	2	x16	2	3	14	10
4	2	2	2	4	x16	2	3	15	10
2	2	2	2	2	x16	2	3	14	10
4	2	2	2	4	x16	2	3	15	10

The following table lists the supported topologies for LPDDR2/LPDDR3 devices. The EMC supports the same topologies on LPDDR2 and LPDDR3 up to 8Gb device density.

Table 50: Supported LPDDR2/LPDDR3 Topologies

DRAM Configuration				DRAM Device Information					
Density (GB)	Number of Channels	Number of Ranks Per Channel	Number of Devices Per Rank	Density (Gb)	Variety	Package	Bank Width	Row Width	Col Width
1	2	1	1	4	S2/S4	x32	3	14	10
2	2	1	1	8	S2/S4	x32	3	15	10
2	2	2	1	4	S2/S4	x32	3	14	10
4	2	2	1	8	S2/S4	x32	3	15	10
1	1	2	1	4	S2/S4	x32	3	14	10
2	1	2	1	8	S2/S4	X32	3	15	10
4	2	2	2	4	S2/S4	X16	3	14	11
2	1	2	1	4	S2/S4	x16	3	14	11
4	1	2	2	8	S2/S4	x16	3	15	11

20.3 Software Features

The majority of software interaction with the memory subsystem involves:

- Initial configuration
- Power management
- Device management
- SMMU translation management
- Surface allocation
- Statistics and debugging

20.3.1 Initial Configuration

The details of configuring the Memory Subsystem (MSS) can be broken down into the following:

- Client configuration
- Protocol Arbiter configuration
- Global Memory Subsystem configuration

> **Note:** This configuration must be performed as part of the chip-wide boot sequence

20.3.1.1 Client Configuration

The primary per-client configuration knob is the target latency. The target latency value is the expected latency hiding of the client. It is statically configured based on the clients request rate and request buffering capability. The value is used by the Memory Subsystem to track the maximum allowed request lifetime in the system.

As an example of client latency tolerance, consider a Display client which has a 32-atom FIFO. If that Display client is expected to handle a 74.25 megapixel per second HDMI interface and is outputting 32-bit RGBA data on that interface, the client's latency tolerance is 32 atoms × 16 bytes per atom ÷ 4 bytes per pixel ÷ 74.25 Mpixels per second = 1.72 µs.

The latency configuration knobs for the system are calculated off a programmable divide-down of the DRAM clock. This divide-down can be changed whenever the DRAM clock is changed, so that the latency knobs always represent the same relative amount of time to the system. For example, at 533 MHz, the divide-down would be programmed to divide by 16 to give a 30 ns tick. If the DRAM speed dropped to 400 MHz, the divide-down would be changed to 12 to maintain the 30 ns tick.

For the above example, the latency timeout would be 1.72 µs ÷ 30 ns = 57 ticks. However, the Memory System does not guarantee that the maximum latency will never be exceeded, so some amount of headroom should be added to this number. Even so, clients should be designed to gracefully handle excessively-delayed transactions; hard-isochronous clients like Display and VI should not require a reset if a pixel does not arrive in time.

In the above scenario, the latency tolerance is use-case dependent. If the Display was doing 16 bits per pixel instead of 32, it would have double the amount of latency. Software can choose to recalculate the latency tolerance, or it can just pick a worst-case value. In general, given an acceptable amount of client buffering, the latency tolerance need not be tuned on a per-use-case basis (but if software does choose to tune the latency, it need only be done within the context of a single driver, since the latency value neither depends on other clients nor on the Protocol Arbiter clock).

Modules like the CPU, which have no well-defined latency requirements (beyond "as fast as possible") can use the latency tolerance value as a tuning knob, trading off performance of the latency-sensitive client versus the overall performance of the memory system. Some tuning of these parameters may be necessary to maximize system performance.

Module designers should choose an appropriate conservative default value for their client's latency tolerance registers. Software should be able to use the default configuration and still meet system use-case targets.

20.3.1.2 Protocol Arbiter Configuration

Each Protocol Arbiter has its own requirements for initialization. Timing parameters specific to the Arbiter, the DRAM and the current operating clock speed have to be written into Protocol Arbiter configuration registers. The type and geometry of the attached DRAM also have to be programmed. Afterwards, the DRAM will require a set of initialization cycles to be issued.

Device Geometry

DRAM devices come in many sizes, widths, etc. The arbiter must be programmed to drive the correct combination of address bits, data bits, and protocol.

Device Timing and Arbiter Timing

A DRAM arbiter has configuration related to timing parameters specific to the attached device. The JEDEC timing specifications for the device have to be converted from nanoseconds to cycle counts. The controller must also be programmed with the cycles per tick number for their particular clock domain (see also the subsection on “Global Memory Subsystem Configuration” below).

Other Arbiter Parameters

Other parameters related to arbitration that need to be configured include:

- overall number of requests outstanding
- which clients are considered isochronous
- which clients participate in power-saving hysteresis operations

Device Initialization and Calibration Cycles

In particular, PAs will need to write registers in the DRAM devices via special cycles to initialize the DRAM. In general all special cycles to a device are done via a hardware/software handshake that looks like:

- Software writes information about the special cycle to an Arbiter-specific register
- The Controller does the special cycle out to the DRAM device(s) when timing requirements are met
- Software checks a status register to ensure the special cycle is complete; or for Mode Register Read operations, the Controller issues an interrupt to indicate that the special cycle has been consumed

Special cycles may be emitted during normal operation as well; the Arbiter will inject them into the data stream at appropriate places (typically along with refresh cycles).

During device initialization, software must ensure that no client can access the memory. This can be done with per-module flush-enable bits.

20.3.1.3 Global Memory Subsystem Configuration

Some parts of the Memory Subsystem must be configured before any transfers are allowed into the system. The address map must be configured to set which portions of the physical address map are allocated to which Protocol Arbiters, and what portions of it are protected by the physical address protection mechanisms. Hardware may enforce address-range-based protection to carve out memory that is only accessible via a Secure TrustZone transaction, or memory that is read-only (to protect OS code). This is in addition to the translations available via the SMMU.

The Arbitration Domains operate off of a unified arbitration clock. This clock is further divided-down to produce a clock-rate independent clock for counting latency intervals across the Memory System (also known as “ticks”). This divide-down should be programmed to a constant interval at initialization (and any other clock-rate change). A 30 ns interval is suggested since it is an even divide-down of many common DRAM clocks, but any convenient granularity may be chosen.

There are also a few global memory system tuning options that tend to shape the memory performance. They are mentioned here for reference:

- The *expiration* value, which is in latency-count units. This is the expected amount of time that the Memory Subsystem thinks it will need on average to "fast-path" a request through the system. A request becomes "hot" when the latency timer reaches the expiration value, and has expired once the latency timeout hits zero[2]. Increasing the expiration value reduces the number of requests that exceed the latency timeout, but at a cost of overall efficiency.
- The *priority inversion limit*, which limits how long a low-priority (non-expired) group of requests can keep a bank open before a high-priority (expired) group of requests will close the bank. The priority inversion limit is in atoms; if a lower-priority page has this many atoms or fewer to retire, it is allowed to keep the bank open and finish its request. If it has more, the bank is closed to allow the high-priority requests and the low-priority requests must re-open the bank later to finish the request group. The priority inversion limit parameter increases worst-case latency while increasing efficiency.
- SMMU configuration. The SMMU will require software to set up and maintain page tables in memory, enable translation for clients, and assign clients to address space identifiers.

20.3.2 Power Management

The primary forms of system power management involve transition to and from low-power DRAM states. The DRAM controller can be programmed to automatically enter a precharge power-down or self-refresh state after some amount of time without a request. The power-down state can be independently entered for each individual DRAM device – it is possible for one device to be in power-down while another is active. If software knows that the system will not be making further DRAM requests, it can cause an immediate transfer to the self-refresh state by writing an override register. If multiple chip selects are available for the DRAM controller, both the hardware and software may choose to enter low-power states on a per-chip-select basis.

Once a request is queued to the controller for that DRAM the hardware will automatically wake from the low-power state if it entered that state via hardware control. This can take a relatively long amount of time since the controller may be required to issue a number of refresh cycles first. For software-initiated self-refresh, software must issue a register write to start the transition from self-refresh back to a fully-powered state.

Deep power-down must be explicitly requested by software, since it causes DRAM data to be lost. Software must disable new incoming requests to the MSS, wait for all outstanding in-flight requests to retire, and then write a register to transition to the deep power-down state. To exit the state, software must reinitialize the DRAM and controller before re-enabling transfers. Deep power-down mode may be enabled for each chip-select in a Protocol Arbiter independently and each Protocol Arbiter independently.

Normal active bank power optimization is managed principally by the Protocol Arbiter. The arbiter will arrange to close banks as soon as possible and to hold requests for as long as the latency timer will allow, with the intent of grouping same-page accesses into a coherent burst. One important case of active power management is the "OSIdle" use case where only Display is active. The Display client can be prone to "dribbling" – that is, sending isolated requests at infrequent intervals.

When doing the OSIdle use case, the Protocol Arbiter will not have enough traffic to keep the client dribble from becoming DRAM dribble (isolated requests that open and close the DRAM pages more often than necessary). To manage DRAM dribble, the controller adds the concept of the "*hysteresis hold-off*", which is a per-controller state bit. When the hold-off is active, the controller is held off from issuing activate commands to open new DRAM pages. Incoming requests will back up in the Protocol Arbiter's row sorter and store buffer until the hold-off is released.

The hold-off is enabled whenever an individual device goes idle due to lack of work. It is released automatically whenever a latency timer expires, or the Protocol Arbiter stalls due to fullness. The net effect of the hold-off is to group *all* requests for the DRAM into activity bursts.

[2] In practice, the latency timeout doesn't actually count down; rather hardware time-stamps the incoming request and computes a deadline value which is based on the current time + latency timeout – expiration. The request has expired when the current time passes the deadline.

The hold-off causes a delay in requests transitioning from a memory-idle state. This can affect some traffic in a negative manner – principally requests that are low-latency like the CPU. To keep the hold-off from impacting these requests, software has a per-module configuration bit that disables the hold-off for certain known-latency-sensitive clients. Whenever a request with the holdoff-disable bit set enters the Protocol Arbiter, the hold-off is released.

A final aspect of power management is scaling the Memory System clocks. Generally there are two sets of clocks that can be controlled:

- The overall Memory System “arbitration clock”, which is the clock used by the Arbitration Domains. Changing this clock reduces the overall bandwidth of the system and reduces the power required by the client interfaces and cross-chip Memory System routing.
- Individual Protocol Arbiter clocks. These reduce the bandwidth available to a particular memory pool and reduce the power consumed by the external pins and the external DRAM. Some Protocol Arbiters may be synchronous to the arbitration clock and not be explicitly controllable via a separate clock enable.

Both clocks may be changed as part of frequency scaling. Changing the arbitration clock rate affects the Arbitration Domain’s idea of the latency timeout. As described in the above section, software should change the divide-down register to keep the scale of latency timeout “ticks” to a (relatively) constant time unit. Generally a tick time of 30 ns is used since it divides evenly into most of the common DRAM frequencies, but any value of tick granularity may be chosen.

Both the arbitration clock and memory clock have separate divide-downs.

Changing the Protocol Arbiter clock rate affects the relative memory timing parameters. The DRAM timing parameters are double-buffered to allow software to update the timing values for the next clock speed setting and then simultaneously switch to using the new values by writing a trigger.

The DRAM protocol itself may require further clock-change restrictions. For example, DDR3 requires a certain number of cycles in which to re-lock the on-DRAM DLL. To further facilitate the hardware/software handoff, and reduce latency for the clock-change operation, a hardware handshake exists between the clock controller module and the Protocol Arbiter.

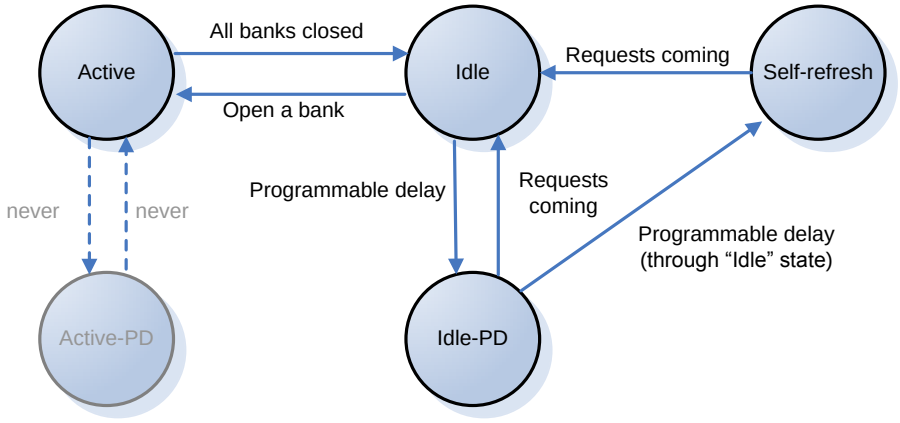
The MSS maintains two copies of each configuration register that may need to change due to a clock-change, the “active” and “assembly” copies; this is also known as a “shadowed” register. The active copy is what the MSS is currently configured to use, and cannot be written to. The assembly copy can be written to at any time, and it can be copied onto the active via a trigger mechanism. The clock-change handshake includes a hardware-initiated trigger for this shadow update.

To assist in full-system power analysis using silicon, the Protocol Arbiter should include enough statistics about the DRAM bus cycles to be able to calculate the DRAM power consumed.

20.3.2.1 Power State Transitions

The power state transitioning is illustrated in the following figure. When all banks are closed, DRAM transitions from active mode to idle mode, from which it can enter idle-powerdown mode. If DRAM accesses are coming, the DRAM will wake up and go back to idle. Otherwise if a long period of time has passed, the DRAM will quit idle-powerdown mode and enter self-refresh mode to save more power. Active-powerdown mode is supported by the EMC but will not happen because the MC does not tell the EMC it is idle until it closes all banks:

Figure 49: DRAM State Transitions and Conditions



20.3.3 Surface Allocation

For linear buffers, software needs to allocate a range of virtual addresses for the buffer and map them to physical pages; the address mapping of bytes within the buffer corresponds to the buffer’s virtual address. For 2D surfaces there are multiple options for layout of the bytes in the surface.

Pitched surfaces are the most straightforward. Assuming 8-bit pixels, for a given pixel coordinate (x, y), the pixel’s byte address is:

Address = Surface_base + y * Surface_pitch + x

For larger pixels, “x” is first multiplied by the number of bytes per pixel.

The Surface_pitch value is the line-to-line stride in bytes of the surface. This may be equal to the surface’s width times the bytes-per-pixel value or it may be additionally padded to a larger stride. Pitched surfaces are conceptually straightforward to use, but they can be inefficient if accesses walk in a vertical or diagonal direction through the surface, since each line may be in a different DRAM row. Software can help on pathologically-sized surfaces (where vertical traversal causes a bank collision) by padding out the Surface_pitch value, but in general they are best used when the surface will only be traversed horizontally or as an interface to naive software that assumes a pitched arrangement.

Tiled surfaces attempt to improve performance for vertical and diagonal traversal by making the DRAM page map more square in the surface. Previous chips used a form of tiling that ordered bytes within the surface into 16x16 tiles:

Address = Surface_base + (y / 16) * (Surface_pitch * 16) + (x / 16) * 256 + (y % 16) * 16 + (x % 16)

The previous memory controller design put the DRAM bank bits at the 1 KB boundary, so a DRAM page ends up being a 64x16 arrangement of bytes. Clients that traverse horizontally get four 16-byte atoms in-page before moving to a new bank, and with a four-bank DRAM get 16 atoms before needing to cycle a bank for a given row, which was enough to ensure efficient access to the page.

Similarly, clients that accessed vertically got 16 atoms in-page before moving outside the tile. Software would still need to pitch-pad the surface to ensure that moving vertically did not cause a bank collision – to do this it needed to make sure that Surface_pitch was not a multiple of 1 KB.

20.3.4 Statistics and Debugging

Statistics are used for performance monitoring and for generating usage information for guiding dynamic voltage and frequency scaling. Hardware will provide a set of statistics counters that are used for DVFS, both for global MSS bandwidth and for individual PA bandwidth. A separate set of counters with additional filtering support will be available for performance monitoring and debugging.

The hardware blocks illegal transfers to memory. Some transfers are based on physical DRAM limitations (addressing outside of the address map), permissions (addressing “secure” physical memory from a non-secure client) and from the SMMU (accessing an unmapped page). All of these are reported via a fault interface which records the transfer information of the last faulting address, including the following data:

- Virtual address of fault
- Module and client ID of fault
- Fault type

Faulting transfers continue through the system, but faulting reads return all-ones and faulting writes discard the write data. Writing a fault triggers a (maskable) interrupt.

20.4 Coherency and Ordering

- The MSS does not guarantee the coherency between different clients.
- The MSS does guarantee that within a single write client, if a write is sent with a write-response tag, all previous writes from that client will complete before returning the write completion tag.
- The MSS also guarantees that within a single client, writes to the same address will complete in-order.

The combination of the three above rules can produce problems if a module accessing memory assumes PCI-like ordering rules; since write ordering is not guaranteed as seen by other clients in the system, it is possible for this scenario to occur:

- Client A is polling a flag location in memory to see that a packet arrives.
- Client B sends write request for packet data to a buffer in memory.
- Client B sends write request to flip the flag to “available”.
- Client B writes to the flag complete.
- Client A sees this and starts reading the packet data.
- Client B’s packet data writes complete.

Obviously in this scenario, Client A reads incorrect data. The recommended way to solve this problem is to avoid polling loops in software as they are inefficient in both performance and power, and instead architect synchronization points in the hardware so that Client B can directly interrupt Client A when it determines all work is complete.

If such modifications cannot be made to the clients, then the writing client must be restricted to one request outstanding. Individual Protocol Arbiters may provide further ordering guarantees. If only one PA exists, then the ordering rules for the MSS may be relaxed to match that of the PA.

For Tegra 4 devices, the SMMU implementation guarantees that the lower 10 bits of the virtual address are untranslated. The Protocol Arbiter implementation guarantees that writes from a single client made to a single page will complete in-order. The Protocol Arbiter also guarantees that the lower 10 bits of the physical address are always column bits. Therefore, Tegra 4 devices can guarantee that writes within the same kilobyte-aligned kilobyte of address will complete in-order, regardless of whether the client is dealing in virtual or untranslated physical addresses.

20.5 Hardware / Software Partitioning

Software needs to be involved in the configuration, translation, and power management services as described above. The main CPUs will run their clients as untranslated by the SMMU, instead relying on the CPU's MMU to do translation. The system does not provide SMMU translation for the CPU clients. It is up to software to keep the CPU and SMMU page tables matched.

Surface allocation is straightforward; software allocates the surface out of the device's virtual space, and then backs the mappings with physical pages. Software should make sure to make the surface mappings not bank collide by pitch-padding surfaces to prevent vertical bank conflicts (generally, this means making sure the pitch for block-linear surfaces is not a multiple of 256 bytes). For surfaces that are accessed in tandem (YUV planar surfaces, Depth and Color for 3D, etc.) software should align the surfaces to start on different banks and use the BANKSWIZ surface kinds to remove page collisions.

20.6 LPDDR3 CA Training and Swizzling

LPDDR3 CA training is required from the latest JEDEC committee conference and is still being defined.

CA training is required at cold boot, and likely not needed on the fly or at warm boot. The CA training sequence is in the Boot ROM. Software can alternatively use micro-boot instead of Boot ROM code to initialize DRAM. The extra delay caused by that at cold boot is not an issue.

Reset, CA training entry, and CA training exit are the only three commands that are – or can be issued as – single data rate. Thus they can be issued at normal frequency without CA training. Resetting after each stepping of training may not be necessary, but it can prevent a training pattern that could be latched as training-exit command by mistake.

Byte swizzling and bit swizzling in bytes are required for CA training. Channel swizzling is requested for DSC 64-bit memory support. The differences between the different swizzling operations are:

- Byte swizzling: the bytes can swap. Bytes are swapped together with their corresponding DQS and DQM signals.
- Bit swizzling: The bits can swap inside any byte.
- Channel swizzling: A specific swizzling is requested for a second method of mapping pin signals. Each channel has three address stacks. You may swizzle within each stack but are not allowed to swizzle among different stacks. Data bytes may be swapped within each channel as well as bits within a byte. DQS/DQM must remain with their byte groups.

For more information on configuring the registers for CA training and swizzling, refer to the EMC_CA_TRAINING_START_0 register description.

20.7 Software Interfaces

The primary software interface for configuration is via the APB bus registers.

- Initial configuration
- Power management
- Device management
- SMMU translation management
- Surface allocation
- Statistics and debugging

The most important tool for configuration is **emc_reg_calc**, which is a tool to generate the DRAM timing and configuration sequences based on information from a vendor's datasheet. Contact your NVIDIA support engineer for further details.

20.7.1 Boot

Tegra 4 devices have a two-stage boot:

- The Boot ROM is responsible for getting the chip out of reset to where driver-level software can take over.
- The Boot Loader or the OS-recovery code is the portion of the driver responsible for completing the boot process to the general-purpose OS.

The Boot ROM is an actual ROM built into the chip, and the Boot Loader and OS recovery section is externally loaded code. The Boot Loader is only used for Cold boot; Warm boot utilizes operating-system recovery code.

Tegra 4 devices can “boot” in two ways: “Warm boot” or “Cold boot”. Cold boot is defined as booting from a fully powered-off or fully reset state. Warm boot is defined as recovery from the lowest-power state (LP0), where the external DRAM has been held in self-refresh mode while the Tegra 4 core was powered off. Warm boot use case occurs much more often; for example, it is the use case hit whenever a SmartPhone wakes from Standby to answer a call.

20.7.1.1 Cold Boot

Before Memory Subsystem boot can begin, the Boot ROM must read the Boot Configuration Tables (BCTs) from flash memory (slow storage) into on-chip RAM. These tables include the configuration data for up to 4 initial DRAM configurations supported. The Boot ROM then reads the strap signals assigned to the SDRAM and begins to bring up the memory subsystem.

The Cold boot Boot ROM sequence:

1. Configure PLL used by the MC/EMC based on BCT values
2. Wait for PLLs to lock
3. Program necessary always-on static pad configurations in the PMC
4. Configure memory clocks based on BCT values
5. Enable the MC/EMC clocks
6. Release memory subsystem resets
7. Program other necessary pad/pad-macro configuration
8. Program initial configuration for the MC/EMC based on BCT values
9. Perform the DRAM initialization sequence. This sequence is specific to the DRAM protocol used and varies depending on the requirements of a particular device. For example, a DDR3 sequence goes as follows:
    a. Apply power to the device
    b. Wait until voltage is stable
    c. Assert DRAM reset
    d. Wait 200 µs
    e. Deassert DRAM reset
    f. Apply stable clocks
    g. Wait 500 µs
    h. Assert and hold CKE high
    i. Precharge all banks
    j. Send 2 Auto-refresh commands
    k. Write to Mode register 0, 1, 2, 3
    l. Issue ZQ Calibration command
    m. Wait for ZQ calibration and DRAM DLL lock time to complete
10. Enable power saving features such as clock stop, active power down, dynamic-self-refresh

11. Start periodic sequences such as auto refresh, ZQ calibration, auto-calibration
12. Release the memory-initialization hardware lock[3]

After this, the external DRAM and Memory Subsystem are ready for use.

20.7.1.2 Warm Boot

The Warm Boot sequence is very similar to the Cold Boot sequence, and many of the same steps are performed. One difference is that instead of powering on the DRAM, we are waking it out of SelfRefresh. Another difference is the configuration data for Boot ROM portion of MC/EMC initialization is stored in the Power Management Controller (PMC) Scratch registers.

The PMC Scratch registers are in the Always-On partition, and are powered during LP0. The power and area limitations of this implementation restricts the number of PMC Scratch registers available; there are a smaller number of PMC Scratch Registers than the number of bits needed to fully configure the MC and EMC. The Scratch Registers are only used to store the minimum required configuration to get to working DRAM. The Boot ROM may not be able to Warm boot to full POR speed using only the Scratch Register data; if this limitation is confirmed, the BootLoader may be required to mimic DVFS software and reprogram the MC/EMC to full-speed operation using data stored in DRAM. It is up to the LP0 Entry software code to ensure the proper configuration gets written to the Scratch Registers.

The Warm boot code provided by the MSS includes algorithms for deriving the approximate MC arbiter performance parameters from the EMC timing parameters. The EMC has provided generation code for the packing/unpacking calls for the scratch registers to ease the software maintenance burden.

The Warm boot Boot ROM sequence:

1. Configure memory PLLs and memory clocks based on Scratch values
2. Wait for PLLs to lock
3. Configure memory clocks based on scratch values
4. Enable the MC/EMC clocks
5. Release memory subsystem resets. Program any necessary static pad configuration that was powered-off in LP0
6. Program initial configuration for the MC/EMC based on Scratch values
7. Release the DRAM from software-controlled Self Refresh. This sequence is specific to the DRAM protocol used and may further vary depending on the requirements of a particular device. For example, a DDR3 sequence goes as follows:
    a. Apply stable clocks
    b. Assert and hold CKE high
    c. Issue ZQ Calibration command
    d. Wait for ZQ calibration and DRAM DLL lock time to complete
    e. Write to Mode registers if different from LP0 entry values
    f. Issue refreshes to ensure tREFI is satisfied
8. Enable power saving features such as clock stop, active power down, dynamic-self-refresh
9. Start periodic sequences such as auto refresh, ZQ calibration, auto-calibration
10. Release the memory-initialization hardware lock

After the Boot ROM sequence is complete, the operating system restore code may access DRAM and restore a different or higher-speed operation. The operating system may also restore translation or client configuration settings before allowing any user code to execute.

[3] AHB_ARBITRATION_XBAR_CTRL.MEM_INIT_DONE

20.7.2 Security Apertures

To secure the MC registers, the ARM® Cortex™-A15 MPCore™ page table entries for the MC register space must be marked as secured. Then to access the registers securely, the Cortex-A15 must be switched into secure mode. For more details about these Cortex-A15 operations, please read the Cortex-A15 architecture manual from ARM.

To secure a region of EMEM, the Cortex-A15's page table entries for that region must be marked as secured. Then the MC registers must be configured for the secured region, using Cortex-A15 in the secured mode:

RegWrite(MC_SECURITY_CFG0, security_aperture_base_address);

RegWrite(MC_SECURITY_CFG1, security_aperture_size_in_megabytes);

After the registers are updated, the Cortex-A15 must be in secure mode to access the described region of memory.

To protect a region of EMEM, the MC register must be configured for the protected region, using Cortex-A15 in the secured mode:

RegWrite(MC_SECURITY_CFG2, carveout_base_address);

For more information on handling security violations, see the subsection "MC Interrupts" below.

20.7.3 Virtual Addressing: Using the SMMU

For a discussion of the virtual-address translation features provided by the SMMU, please read subsection on "Virtual Addressing" below.

In Tegra 4 devices, it is physically impossible to enable SMMU translation for the Cortex-A15 CPU.

There is no way to apply different ASIDs to requests that arrive at the MC through the same MCCIF client. An example would be all three USB controllers access EMEM through the AHBSLV clients on Tegra 4 devices; thus there is no way to assign those USB controllers to different ASIDs.

There is a need on the software side to distinguish between an invalid mapping (which has an invalid PA field) and a mapping that is valid, but has no access. The following pseudo-code describes the recommended conventions.

```
// page/section/address space permission types
//
// note that INVALID implies the PA field is unused.
// PA is valid for all others, including NO ACCESS.
enum Permission {
     INVALID           = NvU32(0),
     NO ACCESS         =                        NONSECURE,
     SECURE WRITE ONLY =            WRITABLE,
     WRITE ONLY        =            WRITABLE | NONSECURE,
     SECURE READ ONLY  = READABLE,
     READ ONLY         = READABLE |             NONSECURE,
     SECURE            = READABLE | WRITABLE,
     ANY               = READABLE | WRITABLE | NONSECURE
 };
```

For more information on handling SMMU faults, see the subsection on "MC Interrupts" below.

20.7.3.1 SMMU Initialization

1. For each Virtual Address Space (VAS) to be used, set up the page table in the DRAM. See the subsection on "Virtual Addressing" below for details on the page-table formats supported.
2. For each ASID, set up the pointer to the page-table base and the ASID-wide protection bits. Since the PTB pointers are stored in a RAM, this uses an indirect access mechanism involving two register writes:

a. Write MC_SMMU_PTB_ASID_0 with the ASID to be modified.
b. Write MC_SMMU_PTB_DATA_0 with the information for this VAS:
    - Page Table Base pointer
    - Whether non-secure accesses are allowed
    - Whether reads are allowed
    - Whether writes are allowed

3. Assign hardware engines (SWNAMES) to the ASIDs and turn on translation for those engines. This involves writes to various MC_SMMU_<SWNAME>_ASID_0 registers.
4. If needed for Hardware Diagnostics purposes, disable or enable translation within a SWNAME on a per-client basis. This involves writes to various bits in MC_SMMU_TRANSLATION_ENABLE_*_0 registers.
5. If desired, secure the ASIDs, which prevent untrusted code from modifying the ASID PTB pointers by enabling a requirement for TrustZone security on register writes that modify the PTBs. This operation involves a write to MC_SMMU_ASID_SECURITY_0. Note that this register itself is secured by TrustZone to prevent it from being modified by untrusted sources, and in some situations writes to it must also be TrustZone secured for them to take effect. In particular, you cannot change a module's ASID with a non-secure write if you are trying to change it to a secure ASID, or if it is already a secure ASID.
6. If desired, enable ASID promotion, which allows clients without security access to inherit the security status based on the page-table translation. This operation involves a write to MC_SMMU_ASID_SECURITY_0.
7. Write PTC_FLUSH_TYPE=ALL to MC_SMMU_PTC_FLUSH_0.
8. Write TLB_FLUSH_VA_MATCH=ALL to MC_SMMU_TLB_FLUSH_0.
9. Configure reserved bits found to be necessary during chip bring-up, if any. There are reserved bits in MC_SMMU_PTC_CONFIG_0 and MC_SMMU_TLB_CONFIG_0.
10. Enable the SMMU by writing SMMU_ENABLE=ENABLE to MC_SMMU_CONFIG_0. Note that this register is secured by TrustZone to prevent it from being modified by untrusted sources, and writes to it must also be TrustZone secured for them to take effect.

20.7.3.2 Page Table Modifications after Initialization

Modifications may be made to the page tables while the system is operating, however, care must be taken to avoid modifying an entry that is in active use. Modifying an entry in active use will result in undefined operation; the most likely failure is the use of an out-of-date cached translation. The general rules for safe modification are:

- If no client is using an ASID, then that ASID and its page table can be modified.
- If no client is using any PTE in a PDE, that PDE can be modified.
- If no client is using a PTE, that PTE can be modified.

It is up to software to ensure these rules are followed when virtual address translations need changed. Once software has updated the page-table or ASID mappings, the final step is to perform PTC and TLB flushes on the modified entries. Note that the granularity of flushing is at the PTE group, which is the number of PTEs that fit in a memory atom. Software can write:

- MC_SMMU_TLB_FLUSH_0 to flush a specific page group, 4MB section, entire ASID or the entire TLB.
- MC_SMMU_PTC_FLUSH_0 to flush a specific PTE or PDE from the PTC

20.7.3.3 Collecting Statistics on SMMU Performance

The TLB and PTC each have hit and miss statistics counters. To enable statistics collection, set MC_SMMU_*_CONFIG_0.*_STATS_ENABLE == ENABLE, where * stands for PTC or TLB. The counters will begin collecting information when enabled. To check the results, read the appropriate registers:

- MC_SMMU_STATS_TLB_HIT_COUNT_0

- MC_SMMU_STATS_TLB_MISS_COUNT_0
- MC_SMMU_STATS_PTC_HIT_COUNT_0
- MC_SMMU_STATS_PTC_MISS_COUNT_0

These counters do not saturate when full; instead they wrap from MAX_UINT to 0. There is no explicit reset for these counters; the only reset-like option is to force the counter bits to all 1's using the MC_SMMU_*_CONFIG_0.*_STATS_TEST trigger bits. This is essentially like resetting the counters to -1.

20.7.4 DVFS Sequences

DVFS (Dynamic Voltage and Frequency Scaling) is a software feature for controlling system power. A key hardware feature for supporting that is changing SDRAM frequency on the fly with minimal impact on system performance. That indicates the following requirements:

- EMC timing parameters and their corresponding MC arbitration timing parameters need to be adjusted to the optimal performance for the new frequency. These parameters can usually be derived from DRAM datasheet and do not need characterization.
- All trimmer values need to be adjusted for the new frequency. These values need to be characterized across PVT (process, voltage, temperature) and frequencies.
- The pad configurations that can be turned off to save power at low frequencies should also change with frequency change, such as pad terminations and DQS pulls.
- All registers involved in frequency change should be shadowed. That includes the reserved register fields that can possible be turned on only in a certain frequency range. Note that any Hardware Diagnostics feature that has power implications should be considered to be shadowed.
- No software interference should be expected from software after software pulls the trigger to change frequency and before the clock change completes: Software runs on CPU from SDRAM. Once the EMC blocks the requests to SDRAM, the CPU may be frozen until the EMC finishes frequency change and unblocks the requests. Note that the CPU can also possibly run out of the contents in cache during clock change, so we need to consider both scenarios.
- The delay caused by clock change should not cause functional failures or visible artifacts.

That result in a design as follows:

- The CAR module handshakes with the EMC on clock change. The EMC handshakes with the MC on shadow register latching. The whole process is done by hardware and transparent to software.
- A rough estimation of clock change delay is 1-4 µs for LPDDR2, 2-6 µs for DDR3.
- Supports two clock change mode: self-refresh clock change mode and power-down clock change mode. Both LPDDR2 and DDR3 support both modes. We recommend clock change power-down mode for LPDDR2 since it takes shorter time. We recommend self-refresh clock change mode for DDR3, since power-down clock change mode does not allow DLL to be turned off.
- For flexibility, EMC has a 16-deep FIFO to hold register programming and release them during or immediately after clock change. That enables us to send DRAM command or latch in unshadowed registers. Here are a few examples:
  - Self-refresh clock change mode requires programming EMC_SELF_REF to put DRAM in self-refresh during clock change.
  - DDR3 precharge-power-down mode requires programming EMC_PRE to close all banks before clock change.
  - Both DDR3 and LPDDR2 clock change requires writing mode registers after clock change.
  - To support DLL-ON/DLL-OFF mode switching on DDR3, program mode registers before or after clock change according to the DDR3 specification.

This is a summary of the frequency change process:

- Software turns off features like self-refresh or auto-calibration if necessary to minimize their interference with the DVFS sequence.
- Software programs MC/EMC shadow registers for the new frequency
- Software programs EMC clock change flow control registers, such as entering self-refresh before clock change and programming SDRAM mode registers after clock change.
- Software programs the MC/EMC clock source register to trigger the clock change.
- The CAR sends clock change request to the EMC (clock divider and/or clock source).
- The EMC stalls all DRAM commands including refreshes.
- The EMC handshakes with the MC to stalls incoming transactions and waits until EMC pipe is flushed
  Within handshaking, EMC may enters power down state if it is in power-down clock change mode.
- The EMC executes pre-clock change sequence such as entering self-refresh and programming mode register to turn off DLL.
- The EMC latches in shadowed registers and requests the MC to latch in shadow registers.
- The EMC acknowledges the CAR to change clock.
- The CAR changes the MC/EMC clock.
- The CAR deasserts the clock change request to the EMC indicating the clock change is done.
- The EMC executes post-clock-change sequence, such as exiting self-refresh and programming SDRAM mode registers.
- The EMC deasserts the clock change acknowledgement to the CAR.
- The EMC unstalls DRAM commands including refresh and resumes incoming MC transactions.
- Software waits 1µs before continuing.

Software programming sequence on the Tegra 4 clock change sequence:

- Before any clock change, usually at boot time:
  - Keep these register fields in reset values:
    CLKCHANGE_REQ_ENABLE = ENABLED
    CLKCHANGE_SR_ENABLE = DISABLED
  - CLKCHANGE_PD_ENABLE:
    ENABLED: power-down clock change mode (for LPDDR2)
    DISABLED: self-refresh clock change mode (for DDR3)
- At each clock change:
  - Clear clockchange_complete flag so we can poll it later for clock change completion:
    EMC_INTSTATUS.CLKCHANGE_COMPLETE_INT = SET
  - Disable DSR (dynamic-self-refresh) if it is currently enabled
    EMC_CFG. DYN_SELF_REF = DISABLE
  - If this clock change is going to switch from SCHMITT to VREF DQ mode, enable VREF DQ:
    EMC_XM2DQSPADCTRL2.EMC2TMC_CFG_XM2DQS_E_VREF_DQ = 1
  - If this clock change is going to switch from SCHMITT to VREF DQS mode, enable VREF DQS:
    EMC_XM2DQSPADCTRL3.EMC2PMACRO_CFG_XM2DQS_E_VREF_DQS = 1
  - If this clock change is going to turn on QUSE IVREF, enable it:
    EMC_XM2QUSEPADCTRL.EMC2TMC_CFG_XM2QUSE_E_IVREF = 1

- Latch shadow registers:
  EMC_TIMING_CONTROL = 1
  Poll for latching completion: EMC_EMC_STATUS_TIMING_UPDATE_STALLED == 0
- Wait 5µs to allow all actions above have finished
- If CFG_XM2COMP_VREF_CAL_EN mode needs to change with the clock change, turn off auto-calibration:
  EMC_AUTO_CAL_INTERVAL = 0
  Then poll for the possible on-going auto-cal to complete:
  EMC_AUTO_CAL_STATUS.AUTO_CAL_ACTIVE == 0
- Program the shadow registers for the new frequency, listed below:
  EMC_RC, EMC_RFC, EMC_RAS, EMC_RP, EMC_R2W, EMC_W2R, EMC_R2P, EMC_W2P, EMC_RD_RCD, EMC_WR_RCD, EMC_RRD, EMC_REXT, EMC_WEXT, EMC_WDV, EMC_QUSE, EMC_QRST, EMC_QSAFE, EMC_RDV, EMC_REFRESH, EMC_BURST_REFRESH_NUM, EMC_PRE_REFRESH_REQ_CNT, EMC_PDEX2WR, EMC_PDEX2RD, EMC_PCHG2PDEN, EMC_ACT2PDEN, EMC_AR2PDEN, EMC_RW2PDEN, EMC_TXSR, EMC_TXSRDLL, EMC_TCKE, EMC_TFAW, EMC_TRPAB, EMC_TCLKSTABLE, EMC_TCLKSTOP, EMC_TREFBW, EMC_QUSE_EXTRA, EMC_FBIO_CFG6, EMC_ODT_WRITE, EMC_ODT_READ, EMC_FBIO_CFG5, EMC_CFG_DIG_DLL, EMC_CFG_DIG_DLL_PERIOD, EMC_DLL_XFORM_DQS0, EMC_DLL_XFORM_DQS1, EMC_DLL_XFORM_DQS2, EMC_DLL_XFORM_DQS3, EMC_DLL_XFORM_DQS4, EMC_DLL_XFORM_DQS5, EMC_DLL_XFORM_DQS6, EMC_DLL_XFORM_DQS7, EMC_DLL_XFORM_QUSE0, EMC_DLL_XFORM_QUSE1, EMC_DLL_XFORM_QUSE2, EMC_DLL_XFORM_QUSE3, EMC_DLL_XFORM_QUSE4, EMC_DLL_XFORM_QUSE5, EMC_DLL_XFORM_QUSE6, EMC_DLL_XFORM_QUSE7, EMC_DLI_TRIM_TXDQS0, EMC_DLI_TRIM_TXDQS1, EMC_DLI_TRIM_TXDQS2, EMC_DLI_TRIM_TXDQS3, EMC_DLI_TRIM_TXDQS4, EMC_DLI_TRIM_TXDQS5, EMC_DLI_TRIM_TXDQS6, EMC_DLI_TRIM_TXDQS7, EMC_DLL_XFORM_DQ0, EMC_DLL_XFORM_DQ1, EMC_DLL_XFORM_DQ2, EMC_DLL_XFORM_DQ3, EMC_XM2CMDPADCTRL, EMC_XM2DQSPADCTRL2, EMC_XM2DQPADCTRL2, EMC_XM2COMPPADCTRL, EMC_XM2VTTGENPADCTRL, EMC_XM2VTTGENPADCTRL2, EMC_XM2QUSEPADCTRL, EMC_XM2DQSPADCTRL3, EMC_CTT_TERM_CTRL, EMC_ZCAL_INTERVAL, EMC_ZCAL_WAIT_CNT, EMC_MRS_WAIT_CNT, EMC_AUTO_CAL_CONFIG, EMC_CTT, EMC_CTT_DURATION, EMC_DYN_SELF_REF_CONTROL, EMC_FBIO_SPARE, EMC_CFG_RSV, MC_EMEM_ARB_CFG, MC_EMEM_ARB_OUTSTANDING_REQ, MC_EMEM_ARB_TIMING_RCD, MC_EMEM_ARB_TIMING_RP, MC_EMEM_ARB_TIMING_RC, MC_EMEM_ARB_TIMING_RAS, MC_EMEM_ARB_TIMING_FAW, MC_EMEM_ARB_TIMING_RRD, MC_EMEM_ARB_TIMING_RAP2PRE, MC_EMEM_ARB_TIMING_WAP2PRE, MC_EMEM_ARB_TIMING_R2R, MC_EMEM_ARB_TIMING_W2W, MC_EMEM_ARB_TIMING_R2W, MC_EMEM_ARB_TIMING_W2R, MC_EMEM_ARB_DA_TURNS, MC_EMEM_ARB_DA_COVERS, MC_EMEM_ARB_MISC0, MC_EMEM_ARB_RING1_THROTTLE
- If this clock change is going to enable DDR3 DLL (and previous frequency disables it), program the time needed for DLL latching into MRS_LONG_WAIT_CNT. You can minus the time that DLL locking overlaps with other commands (such as ZQ-long calibration time, if it is inserted in post clock change sequence below) to reduce this latency.
  EMC_MRS_WAIT_CNT_MRS_LONG_WAIT_CNT = 512 – overlapping-clocks-with-other-commands
- Program STALL_THEN_EXE_BEFORE_CLKCHANGE = 1
  All following EMC register programming will happen after flushing DRAM requests and stalling further DRAM commands, including auto-refreshes.
  IMPORTANT NOTE: EMC register read still stop functioning from this point until the clock change completes: issuing an EMC register read in this duration will hang the system.
- (optional) Program unshadowed EMC registers that need to change with clock change. For example, if periodic_qrst needs to be enabled during clock change, it can be done here. If that register is shadowed, use WRITE_MUX = ACITVE before writing the register, then restore WRITE_MUX after writing the register.

- Disable auto-refresh to reduce the latency of clock change:
  EMC_REFCTRL.DEVICE_REFRESH_DISABLE = depending on DRAM configuration
  EMC_REFCTRL.REF_VALID = DISABLE
- (self-refresh clock change mode only, DLL-ON to DLL-OFF mode switching only, DDR3 only)
  Program EMC_MRS to disable DLL.
- (self-refresh clock change mode only) Program SELF_REF to enter self-refresh:
  SELF_REF.DISABLED = ENABLED
  SELF_REF.DEV_SELECTN = depending on DRAM configuration
- Program STALL_THEN_EXE_AFTER_CLKCHANGE = 1
  All following EMC register programming will happen after clock change but before unblocking DRAM requests
- (optional) Program other unshadowed registers that need to be programmed during self-refresh, such as EMC_XM2CLKPADCTRL register. If periodic_qrst is enabled in clock change but not meant for the new frequency, the value can be restored here.
- (self-refresh clock change mode only) Program SELF_REF to exit self-refresh
  EMC_SELF_REF.DISABLED = DISABLED
  EMC_SELF_REF.DEV_SELECTN = depending on DRAM configuration
- Program DRAM mode registers by MRS/EMRS (DDR3) or MRW (LPDDR2) to change CL/WL/TWR etc.
  On DLL-OFF to DLL-ON switch on DDR3, DLL is turned on with this step. Note that if a mode register has the same value at both the old and new frequencies, it does not have to be programmed.
- Issue ZQ calibration if the clock change is switching from a ZQ calibration disabled frequency to a ZQ calibration enabled frequency.
- Program UNSTALL_RW_AFTER_CLKCHANGE = 1
  Note that all following EMC register programming will happen after clock change, possible with on-going SDRAM access
- Read and discard an arbitrary MC register (Note: cannot use an EMC register) to ensure the register writes are complete.
- Do clock change. Note that until this step, SDRAM access were still happening in the system
  Program CLK_SOURCE_EMC to change clock:
  EMC_2X_CLK_SRC: MC/EMC clock source
  USE_PLLM_UD: use low jitter clock
  MC_EMC_SAME_FREQ: drive EMC vs. MC clock ration at 1:1, otherwise 2:1
  EMC_2X_CLK_DIVISOR: EMC clock divider
- Poll for clock change completion:
  EMC_INTSTATUS.CLKCHANGE_COMPLETE_INT == 1
- Re-enable auto-refresh
  EMC_REFCTRL.DEVICE_REFRESH_DISABLE = depending on DRAM configuration
  EMC_REFCTRL.REF_VALID = ENABLE
- Re-enable auto-calibration if it was previously enabled and then temporarily disabled during clock change:
  EMC_AUTO_CAL_INTERVAL = new value
- Re-enable DSR (dynamic-self-refresh) if it was previously enabled
  EMC_CFG. DYN_SELF_REF = ENABLE
- If EMC_ZCAL_WAIT_CNT was changed before clock change for issuing ZQ-long calibration command, restore the value:
  EMC_ZCAL_WAIT_CNT = new value
- Wait 2 µs for the actions above to complete

- Update shadow registers
  EMC_TIMING_CONTROL = 1
  Poll for latching completion: EMC_EMC_STATUS_TIMING_UPDATE_STALLED == 0

20.7.5 ZQ Calibration Sequence

ZQ calibration is a feature of LPDDR2 and DDR3 to calibrate R on (the output driver) across PVT (process, voltage, temperature), and also ODT values on DDR3. The EMC supports periodically sending ZQ calibration commands to both DRAMs, requiring only initial configuration after booting. The EMC also supports one time calibration commands at booting.

For cold booting, the boot ROM does a ZQ-init on LPDDR2 and a ZQ-long on DDR3, and then enables periodic ZQ-short.

For warm booting, boot ROM does ZQ-long on both LPDDR2 and DDR3, and then enables periodic ZQ-short.

One time calibration:

- LPDDR2, Write MC_MRW:
  - MRW_DEV_SELECTN: 2 for device 0, 1 for device 1. Or use 0 for both (not used because software must control the delays between commands in boot ROM).
  - MRW_MA: 10
  - MRW_OP: 0xFF for ZQ-init. 0xAB for ZQ-long, 0x56 for ZQ-short
- DDR3: Write EMC_ZQ_CAL
  - ZQ_CAL_DEV_SELECTN: 2 for device 0, 1 for device 1. Or use 0 for both (not used because software must control the delays between commands in boot ROM)
  - ZQ_CAL_LENGTH:
    SHORT: issue ZQ-short command
    LONG: issue ZQ-long command
  - ZQ_CAL_CMD: 1

Periodic calibration:

- EMC_ZCAL_INTERVAL: Number of microseconds to wait between periodical ZQ commands. Program this register to 0 to disable periodic ZQ calibration
- EMC_ZCAL_WAIT_CNT: Number of clocks to wait after each ZQ command before other commands
- EMC_ZCAL_MRW_CMD:
  - ZQ_MRW_DEV_SELECTN
    2 for device 0 only, 1 for device 1 only, 0 for both devices
  - ZQ_MRW_MA
    LPDDR2: 10
    DDR3: 0 for ZQ-short, 1 for ZQ-long
  - ZQ_MRW_OP
    LPDDR2: 0x56 for ZQ-short; 0xAB for ZQ-long
    DDR3: 0

20.7.6 MRR Sequence

Mode Register Read (MRR) is desirable on LPDDR2 for reading the manufacturer ID to identify DRAM vendors, or reading back from the LPDDR2 temperature sensor.

Simple sequence:

- Before any MRR, such as at booting:
  - Set MRR_BYTESEL and MRR_BYTESEL_X16 corresponding to data byte sizzling, if applied on board. The reason is that LPDDR2 always returns MRR at low byte of a data strobe, which is the only case that EMC needs to know about swizzling.
- For each MRR
  - Check EMC_STATUS.MRR_DIVLD = 0. If not true, read EMC_MRR until it becomes true
  - Issue MRR:
    MRR_MA: the index of the mode register to read back
    MRR_DEV_SELECTN: 2 for device 0, 1 for device 1. 0 or 3 are illegal. If both devices are desired, repeat this sequence on each device.
  - Poll for EMC_STATUS.MRR_DIVLD = 1
  - Read EMC_MRR. Data is in the MRR.DATA field

20.7.7 Auto-Calibration Sequence

Auto-calibration is for periodically calibrating the output driver of Tegra 4 pads. The EMC supports periodic auto-calibration without software interference, usually initiated at booting.

To enable auto-calibration:

- Program AUTO_CAL_INTERVAL: the number of microseconds between two auto-calibrations
- Program EMC_AUTO_CAL_CONFIG:
  AUTO_CAL_START = 1
  AUTO_CAL_ENABLE = 1
  AUTO_CAL_OVERRIDE: 1 to enable override, 0 to use normal auto-calibration
  AUTO_CAL_PD_OFFSET: override value
  AUTO_CAL_PU_OFFSET: override value
  AUTO_CAL_E_CAL_UPDATE: number of EMC clocks E_CAL_UPDATE is asserted
  AUTO_CAL_STEP: number of microseconds between each calibration step

To disable auto-calibration:

- Program AUTO_CAL_INTERVAL to 0

To re-enable auto-calibration after disabling it

- Program AUTO_CAL_INTERVAL to the new value

20.7.8 LP0/LP1 Entry/Exit

LP0 and LP1 are both power saving modes with DRAM in self-refresh mode. The only difference is that the MC and the EMC are ON during LP1 but powered OFF during LP0.

LP0 and LP1 entry share the same sequence:

- Disable ZQ calibration: EMC_ZCAL_INTERVAL = 0
- Disable Auto calibration: EMC_AUTO_CAL_INTERVAL = 0
- Disable DSR (Dynamic self-refresh) EMC_CFG. DYN_SELF_REF = DISABLED

- Latch shadow registers:
  EMC_TIMING_CONTROL = 1
  Poll for latching completion: EMC_EMC_STATUS_TIMING_UPDATE_STALLED == 0
- Wait 5 µs for worst case DSR exit time
- Program EMC_REQ_CTRL to stall all transactions
  STALL_ALL_WRITES = 1
  STALL_ALL_READS = 1
- Wait until the EMC pipe is flushed
  poll EMC_EMC_STATUS.NO_OUTSTANDING_TRANSACTIONS == 1
- Enter self-refresh
  EMC_SELF_REF.SELF_REF_CMD = 1
  EMC_SELF_REF.SREF_DEV_SELECTN: 2 for device 0 only, 1 for device 1 only, 0 for both
- Wait until the device is in self-refresh
  poll EMC_EMC_STATUS.DRAM_IN_SELF_REFRESH ==
  1: dev0 is in self-refresh; 2: dev1 is in self-refresh; 3: both devices are in self-refresh
- Put VTTGEN in lowest power mode:
  EMC_XM2VTTGENPADCTRL.EMC2TMC_CFG_XM2VTTGEN_DRVUP = 0
  EMC_XM2VTTGENPADCTRL.EMC2TMC_CFG_XM2VTTGEN_DRVDN = 0
  EMC_XM2VTTGENPADCTRL2.EMC2TMC_CFG_XM2VTTGEN_E_NO_VTTGEN = 7
- Latch shadow registers:
  EMC_TIMING_CONTROL = 1
  Poll for latching completion: EMC_EMC_STATUS_TIMING_UPDATE_STALLED == 0
- Put pads in lowest power mode (must program in a single register write):
  APBDEV_PMC_IO_DPD_REQ.CODE = DPD_ON
  APBDEV_PMC_IO_DPD_REQ.POP_ADDR_CMD == ON
  APBDEV_PMC_IO_DPD_REQ.POP_CLK = ON
  APBDEV_PMC_IO_DPD_REQ.DDR_ADDR_CMD = ON
  APBDEV_PMC_IO_DPD_REQ.DISC_ADDR_CMD = ON
  APBDEV_PMC_IO_DPD_REQ.DDR_DATA = ON
  APBDEV_PMC_IO_DPD_REQ.COMP = 1
- Now it is safe to disable PLL, which will freeze EMC and DRAM clocks

LP0-exit is also called warm boot. Refer to the “Warm Boot” subsection.

LP1-exit:

- Enable PLL and waits until clocks stabilize
- Restore pad power state to normal. Note that writing back the original value to this register does not guarantee restoring to the original state. Instead, read and save the value of APBDEV_PMC_IO_DPD_STATUS before LP1 entry (which has value 0 on the register fields representing the enabled pads), and in this step write the inverted value of it (so that it has 1 on register fields representing the same enabled pads) to APBDEV_PMC_IO_DPD_REQ with the field “CODE” set to “DPD_OFF” (meaning “turn it on”).
- Restore the registers to original values before LP1 entry. The register values need to be read and saved before LP1 entry and restored here.
  EMC_XM2VTTGENPADCTRL
  EMC_XM2VTTGENPADCTRL2
  EMC_AUTO_CAL_INTERVAL (this will re-enable auto-calibration)
- Relock DLL.
  Read and write back EMC_CFG_DIG_DLL with field DLL_RESET set to 1.

- Latch shadow registers:
  EMC_TIMING_CONTROL = 1
  Poll for latching completion: EMC_EMC_STATUS_TIMING_UPDATE_STALLED == 0
- Do a single auto-calibration:
  Read and write back EMC_AUTO_CAL_CONFIG with field AUTO_CAL_START set to 1
- Wait for the auto-calibration to complete:
  EMC_AUTO_CAL_STATUS.AUTO_CAL_ACTIVE == 0
- Exit self-refresh
  EMC_SELF_REF.SELF_REF_CMD = 0
  EMC_SELF_REF.SREF_DEV_SELECTN = 2 for device 0 only, 1 for device 1 only, 0 for both
- poll for self-refresh completes
  poll for EMC_EMC_STATUS.DRAM_IN_SELF_REFRESH == 0
- Issue a ZQ calibration:
  LPDDR2:
  1. Write EMC_MRW with
     MRW_DEV_SELECTN = 2
     MRW_MA = 10
     MRW_OP = 0xAB
  2. Wait 1 µs
  3. If this is a two rank configuration, repeat step 1 with MRW_DEV_SELECTN = 1, then repeat step 2

  DDR3:
  1. Write EMC_ZQ_CAL with
     ZQ_CAL_DEV_SELECTN = 2
     ZQ_CAL_LENGTH = LONG
     ZQ_CAL_CMD = 1
  2. Wait 10 µs
  3. If this is a two rank configuration, repeat step 1 with ZQ_CAL_DEV_SELECTN = 1, then repeat step 2
- Program EMC_REQ_CTRL to unstall all transactions
  STALL_ALL_WRITES = 0
  STALL_ALL_READS = 0
- If ZQ calibration was enabled before LP1 entry, re-enable it.
  EMC_ZCAL_INTERVAL = saved value
- If DSR was enabled before LP1 entry, re-enable it:
  EMC_CFG. DYN_SELF_REF = ENABLED
  Latch in the shadow by EMC_TIMING_CONTROL.TIMING_UPDATE = 1
  The poll for latching completion: EMC_EMC_STATUS_TIMING_UPDATE_STALLED == 0

20.7.9 Performance Tuning

The Tegra 4 MSS is architected in such a way that, other than the per-client latency allowance setting, only minimal performance tuning is required.

> **Note:** The capitalization in the below sections refers to the actual register name of the associated setting.

20.7.9.1 Arbitration Knobs

The suggested configuration values produced by the emc_reg_calc program are the best-known values, software should use these values.

The Arbitration Domain provides a set of arbitration knobs that affect the entire MSS, particularly the fairness algorithm in the snap-arbitration. These knobs are:

- RING1_THROTTLE
- RING3_THROTTLE
- Per-partition SNAP_LEVELS

The Protocol Arbiter provides a set of arbitration knobs that affect the entire PA. These knobs are:

- MAX_OUTSTANDING count, and the related controls
- TURN and COVER costs for the Direction Arbiter
- PRIORITY_INVERSION thresholds
- BC2AA_HOLDOFF threshold
- EXPIRING_SOON threshold
- REFRESH_ACK threshold
- DEADLOCK_PREVENTION_SLACK threshold

It is *not* expected that software will need to change these values for best performance; the emc_reg_calc script has been used in the performance verification effort.

20.7.9.2 Per Client Knobs

Each client of the MSS has some configuration knobs that affect the behavior of the request in the PA. These knobs are:

- ISOchronous
- HYSTeresis
- Latency ALLOWANCE

Most clients set their ISO and HYST bits based on the behavior profile of the client; which is static for all systems. The default programming should be acceptable for these bits. The Latency Allowance may need to be programmed on a per-use case basis for some clients.

20.7.9.3 Activity indication for DVFS

The Tegra 4 MSS does not maintain statistics information for DVFS. However, it does send two signals to the centralized Statistics Monitor for that unit that maintains the threshold and interrupts that are used by DVFS. The number of atoms between toggles for these signals can be controlled by ATOMS_PER_DVFS_PULSE.

20.7.9.4 Statistics

Memory Subsystem statistics are for debugging/profiling only. There are two sets:

- MC statistics provide ability to view bandwidth/latency/per-client information.
- EMC statistics provide the ability to view DRAM commands, which you can then use to calculate approximate DRAM power consumption.

20.7.10 Errors (Interrupts)

Both the MC and the EMC implement a handful of interrupts to inform the software of completed events or error conditions encountered by the hardware. Both the MC and the EMC have implemented the same style of interrupt interface, so this section will use the MC as the example.

The interrupt interface for MC consists of two registers and a hardware output wire. The output signal is wired to the central interrupt handling module on Tegra 4 devices; from there it can be routed to any of the CPU complexes. The first register,

INTSTATUS, contains the interrupt vector for MC. Each bit in this register is set when the hardware detects an interrupt. Writing a 1 to the interrupt vector bit will clear the associated interrupt. The second register is INTMASK, each bit in this register corresponds to a vector bit in INTSTATUS. If the MASK bit for an interrupt is *clear (MASKED)*, the corresponding interrupt will *not* forward the interrupt to the central interrupt handling module. If any UNMASKED interrupt vector bits are set, the MC will assert the interrupt to the central interrupt handling module.

20.7.10.1 MC Interrupts

The MC contains two classes of interrupts: address decode errors and performance warnings.

There is currently only one performance warning type interrupt: ARBITRATION_EMEM. It fires when the MC detects that a request has been pending in the Row Sorter long enough for to hit the DEADLOCK_PREVENTION_SLACK_THRESHOLD. In addition to true performance problems, this interrupt may fire in situations like clock-change where the EMC backpressures pending traffic for long periods of time. This interrupt is intended to help developers identify and debug performance issues and configuration issues.

There are three types of address decode errors implemented for Tegra 4 devices, discussed below. All the decode errors share a single pair of information-capture registers that are intended to assist developers in debugging such errors (ERR_STATUS and ERR_ADR). The capture registers record the following information about the error:

- the Virtual or Physical address of the error (depending on the type of fault)
- which type of fault the captured error corresponds to
- a read/write indicator
- the requesting client's ID
- If the error was an INVALID_SMMU_PAGE, then information about the page's protection status is also captured:
  - whether the page was marked non secure in the page-table
  - whether the page was marked readable in the page-table
  - whether the page was marked writeable in the page-table
  - Note that SMMU page protection bits are formed by ANDing the page protection bits from the Page Table Base, PDE and PTEs from all three stages of the page-walk. The ANDed protection bits and the type of the provoking transaction are both available in the status register.

Subsequent errors (of any type) will not change the status and address registers until the corresponding interrupt is cleared.

To prevent requests with address decode errors from modifying memory or accessing memory they do not have permission to, the MC "squashes" the requests. A write request that is "squashed" has had its byte-enables forced to all-zeros, this prevents the write data from being applied to DRAM. A read request that is "squashed" will have its read-return data forced to all-ones, this protects the data in DRAM from being read by non-secure sources.

The three types of address decode errors are:

- DECERR_EMEM: Decode Errors: there is no DRAM attached at the Physical Address.
- SECURITY_VIOLATION: Legacy Security Apertures: the Physical Address is in one of the Legacy Security Apertures, and the client permissions are not sufficient to allow access. There are two Legacy Security Apertures which decode as different ERR_STATUS.ERR_TYPE.

  Note: If a single request causes both a TrustZone and Protected security violation, the ERR_TYPE will report the Protected Aperture violation.

- SMMU_FAULT: The SMMU translation engine encountered either a security permission error, a read/write type error, or an invalid translation error during translation.

SMMU-Translated Requests and Physical Aperture Errors

All Virtual Addresses are translated through the SMMU, then run through the Physical Aperture checks. There are a number of confusing scenarios when debugging these interactions including:

- SMMU fault followed by any physical aperture error of any sort: because the SMMU translation occurs first, the data captured by the ERR_ADR and ERR_STATUS registers will always be the SMMU fault information. Both interrupt bits are set.
- Decode error on a successfully translated (i.e., no SMMU fault indicated) client request: the PA checks will throw a decode error; the data captured includes the Physical Address, not the Virtual Address. The client request is squashed. The decode-error interrupt bit is set.
- Decode error on a Page-Table fetch (PDE or PTE): the page-table fetch will throw a decode error; the data captured includes the Physical Address of the PDE or PTE, and the client ID corresponds to the PTC. In addition, the data returned to the PTC is modified to turn off all access permissions, marking the page INVALID. This forces the client request to also throw a SMMU fault when it is translated. Both the decode-error and SMMU-fault interrupt bits are set. The PDE or PTE are then cached with those modified access permissions, so any subsequent requests to that virtual page are also SMMU fault.
- TrustZone security violation on a translated client request: the PA checks throw a security error, the data captured includes the Physical Address, not the Virtual Address. The security violation interrupt bit is set. The client request is squashed. This behavior allows the Secure Aperture to stand as a “last line of defense” against SMMU page-table modification attacks.
- Protected-region security violation on a translated client request: the PA checks throw a security error. The data captured includes the Physical Address, not the Virtual Address. The security violation interrupt bit is set. However, the client request is not squashed. Due to this behavior, it is not recommended that the Protected Aperture be used in conjunction with SMMU translation.

20.7.11 Software Client-Groupings

The Memory Subsystem inherently implements a hardware mapping of memory clients (“MCCIFs”) to super-clients, and super-clients to the hardware modules. Additionally, the MSS implements a system of mapping modules to “software groups” (also known as SWGROUP or SWNAME). This allows a single point of control for all clients for that hardware “engine”. Often the list of clients in a SWGROUP is the same as the list of clients in the module that implements the hardware engine.

The mapping of SWGROUP to modules for Tegra 4 devices is listed here:

SWGROUP	Modules	Translatable?	Further description
AFI	PCIE	Yes	PCI Express
AVPC	AVP Cache	Yes	Arm7 Audio-Video Processor (AVP)
DC	Display, 1st instance	Yes	Display reads
DCB	Display, 2nd instance	Yes	Display reads, instance b
EPP	EPP	Yes	Encoder Pre-Processor
G2	GR2D	Yes	2D engine
HC	Host1x	Yes	Host interface
HDA	HDA	Yes	High-definition Audio engine
ISP	ISP	Yes	Image Signal Processor
MPCORE	AXICIF	No	Cortex-A15
MPCORELP	AXICIF_LP	No	Cortex-A15
MPE	MPEA, MPEB, MPEC	Yes	MPEG4 Encoder
NV	TEX, IDX, FDC	Yes	3D engine, 1st instance
NV2	TEX, IDX, FDC	Yes	3D engine, 2nd instance
PPCS	AHBDMA, AHBSLV	Yes	Clients in the AHB cluster.

SWGROUP	Modules	Translatable?	Further description
PTC	MC (SMMU)	N/A	Misses from SMMU PTC.
SATA	SATA	Yes	Serial ATA.
VDE	VDE	Yes	Video Decode Engine
VI	VI	Yes	Video Input engine.

20.8 Functionality

20.8.1 Addressing

The physical address space supported by the MC is 4GB. Within this 4GB range, one physical aperture and two virtual apertures are defined. The physical aperture maps external memory. The security aperture allows software to define a region of external memory as secure so that only trusted clients[4] can access that region. The protected aperture allows software to define a region of external memory as carve-out so that only certain clients can access that region.

For any particular chip, the size and locations of these apertures are fixed based on project specific definitions. For Tegra 4 devices, these apertures are defined to be: 2GB maximum for external memory, up to 2GB secured region, up to 2GB protected region.

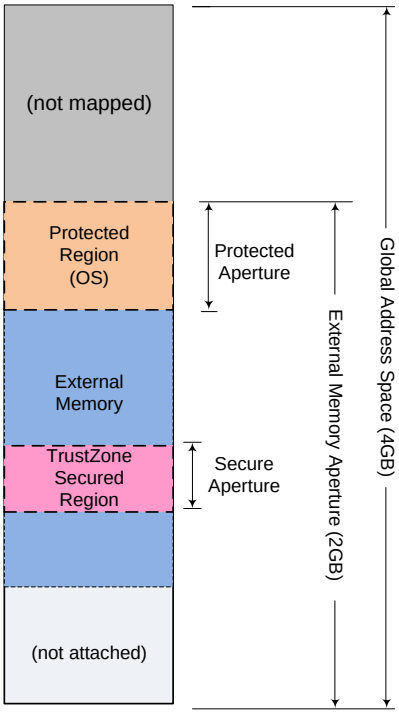
In addition to the physical address apertures and spaces, the MC supports a full virtual addressing scheme via the SMMU.

The MC receives a full 32-bit physical address and decodes it in the following way:

- The external memory is present in a fixed aperture of 2GB at a naturally aligned boundary defined by the system address map. The actual size of the external memory is defined by a programmable register; addresses outside the programmed range are treated as invalid. The offset in the external memory is given by the 31 LSBs of the address.
- A secured region of variable size can be defined in the external memory range.
- A protected region of variable size can be defined in the external memory range, but always starts from offset 0x0.

[4] See subsection on TrustZone Security Region below for a discussion of security.

Figure 50: Physical Address Map Example



Accesses that do not match one of the three address spaces are detected and an error logged. The MC returns a predefined pattern[5] for reads to invalid addresses and acknowledges invalid write addresses, but otherwise the EMC drops the write.

20.8.2 Granularity

The access granularity[6] inside the MC is 128 bits (16 bytes) for external memory accesses. Byte access remains possible but involves a minimum granularity access. Write accesses can use byte enables to write a set of bytes (including a null set), read accesses always return a full word with potentially unneeded data (also known as overfetch).

The granularity of external access is linked to the burst size programmed for DRAM access. Bursts into external memory always start at a 16-byte aligned boundary. For LPDDR2, this corresponds to a burst length of 4 data transfers (or 2 clock cycles at double data rate, known as BL4).

The DDR3 situation is complicated by the DRAM's minimum internal burst length of 8; because the MSS works in 16-byte atoms, a naïve implementation using BL4 or BL8 setting would result in 50% efficiency loss. However, the DDR3 specification also includes a burst-chop on-the-fly mode that allows the controller to indicate for each CAS command whether it wants to transfer 4 or 8 data beats. The BL4 or BC4 accesses still consume a full 8 data beats, but some amount of power is saved by not transferring the data off-chip. The MC and EMC implement support for coalescing of 2 atoms into a 32-byte-aligned pair.

[5] All data bits are forced to 1's.

[6] A datum of 128-bit-granularity may be referred to as a "memory atom".

20.8.3 Virtual Addressing

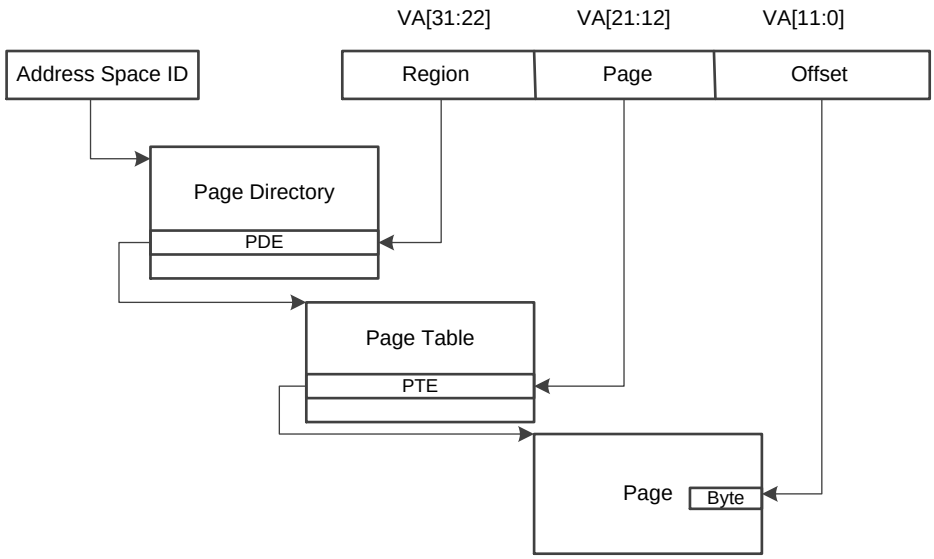
The SMMU is a centralized virtual-to-physical translation for MSS. It allows the following:

- Memory allocation for hardware pools (surfaces and buffers) could be made from non-contiguous page-sized chunks of memory
- Areas of the hardware memory map could be marked as protected, preventing access from hardware devices
- Hardware memory access can be controlled on a per-device or per-process basis

The virtual-to-physical translation uses a two-level page table that carves up the virtual address as follows:

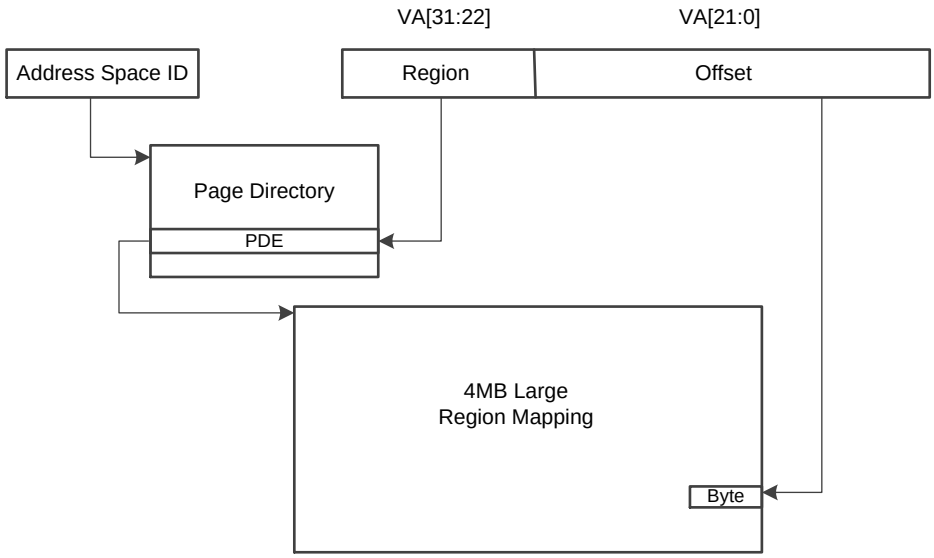
- Bits 31:22 choose a page directory entry (PDE). Each page directory has 1,024 PDEs, each mapping 4MB region.
- Bits 21:12 choose a page table entry (PTE) from a page table. Each page table has 1,024 PDEs, each mapping a 4kB page.
- Bits 11:0 choose a byte within a page

Figure 51: Page Table and Virtual Address



Each PTE or PDE consumes 32 bits of memory, so that each page directory or page table consumes one 4kB memory page. In addition, the page directory lookup does an early-out either to map an entire 4MB region of the virtual space as invalid, or to map the region to a contiguous range of physical pages.

Figure 52: Large Region Mapping



Both the page and directory tables are stored in DRAM. A translation look-aside buffer (TLB) is used to cache recently used translations. Translations that miss in the TLB incur potentially two additional round-trip DRAM delays, but the latency-scheduling features of the Protocol Arbiter is used to hide this latency behind the normal latency incurred by the DRAM reordering policy. The TLB is backed by a page-table cache (PTC) to reduce this memory traffic further.

In addition, the SMMU supports multiple address spaces, each tagged with an address space identifier (ASID). The ASID allows for multiple virtual mappings to exist at the same time. Typically, software uses these to assign different mappings to each process that directly uses the hardware drivers, allowing process isolation. The number of ASIDs in a Tegra 4 device is 4. Software is responsible for managing the assignments between processes and ASIDs, flushing ASIDs out of the TLB as needed when they are remapped between processes.

ASIDs are assigned based on the SWNAME of the incoming request. An on-chip mapping between SWNAME and the ASID is maintained by the SMMU. For register-based devices, the driver is responsible for updating the mapping between the SWNAME and the ASID when a device in context is swapped between processes.

Untranslated modules are also supported. The SWNAME-to-ASID mapping has a translation-enable bit to allow translation on a per-SWNAME basis. There is also a per-client translation-enable bit for Hardware Diagnostics purposes.

Typically, only the CPU uses the untranslated mode, since it has its own MMU and can map any physical page into a process at will. Untranslated accesses for all clients are fast-pathed through the MC/SMMU system so that the latency of such accesses are not impacted more than necessary. To further improve CPU latency, the Tegra 4 CPU low-latency read paths bypass the SMMU altogether; the hardware also restricts the CPU write paths to untranslated-mode to prevent any configuration mishaps. In addition, a global “translation enable” bit disables all translation when cleared (cleared is the default out of reset).

20.8.3.1 Page Table Layout

The in-memory layout of a PTE looks like

- 20 bits of physical address (PA[31:12]) for the base of the page
- Protection bits which include
  - A “readable” bit which allows clients to read this page
  - A “writeable” bit which allows clients to write this
  - A “nonsecure” bit which (when unset) restricts access of this page to “secure” TrustZone clients

The remaining bits in the 32-bit word are reserved.

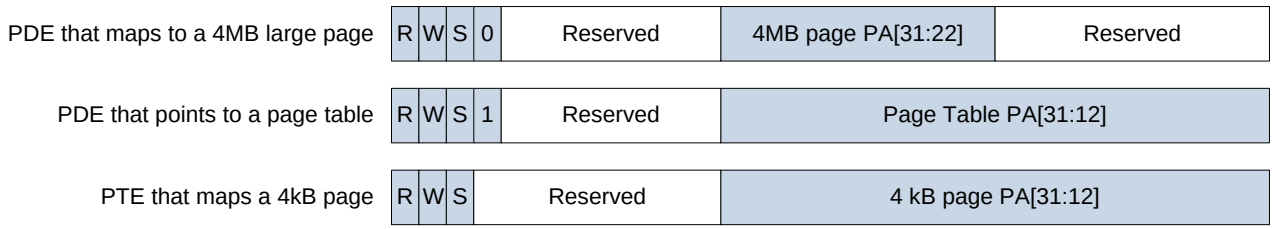
> **Note:** Wherever physical addresses are used in this document, it is assumed that the device can address 4GB of physical memory. The actual physical memory that can be attached to the device may be smaller or larger, and if so, any of the hardware structures that hold physical addresses shrink or grow accordingly. This means that, for example, in a 2GB system, the physical addresses in the PDE and PTEs have only a 19 bit field, and the TLB and PTC need only cache those 19 bits instead of the full 20.

The in-memory layout of a PDE looks like:

- A “next level” bit which says if this PDE maps a single 4MB region, or if it points to a page containing a table of PTEs for the second level of translation
- If the “next level” bit is set, there are 20 bits of physical address (PA[31:12]) that point to a page containing the base of the PTE table for this PTE. If the “next level” bit is clear, we use 10 bits (PA[31:22]) to point to the base of an aligned 4MB physical region mapped directly by the PDE.
- The remaining bits set the permissions for the access. For 4MB large pages, these are used directly for the region. For PDEs that point to a second-level page table, these permissions are ANDed with those of the PTE to give the final permission
  - A “readable” bit for the region
  - A “writeable” bit for the region
  - A “nonsecure” bit for the region

The remaining bits in both interpretations are reserved.

Figure 53: PDE and PTE Formats



The base address of the page directory for a particular ASID is stored on-chip in the ASID table. Changes to this table may require flushing the ASID in question from the TLB.

20.8.3.2 Faults

When an access violates the “readable”, “writable” or “nonsecure” bits for the region, it generates a fault. A fault can cause an interrupt to be generated if desired. In addition, information about the most recent fault is stored on chip, consisting of the following:

- VA[31:4] of the faulting access
- Access type (read/write, secure/non-secure)
- Client ID
- The page permissions at the time of fault (readable, writeable, nonsecure)

Only faults for translated requests use this mechanism. Untranslated requesters (including the SMMU itself) report other types of faults (address out of range, etc.) via the existing MC fault reporting mechanism.

20.8.3.3 Flushes and Page Table Updates

Changes to the page tables require the TLB to be flushed. A software interface for selectively flushing parts of the TLB is provided. The flush command can flush any TLB entries that match a particular subset of virtual addresses. Each address component may be included for matching:

- Match on the ASID, or not
- Match VA[31:24], or Match VA[31:14], or no VA match

Disabling all matches flushes the entire TLB. Note that the TLB is 4 entries wide, so when you match any particular VA you flush the entire line, which is why the VA matches are down to 24 and 14, rather than 22 and 12.

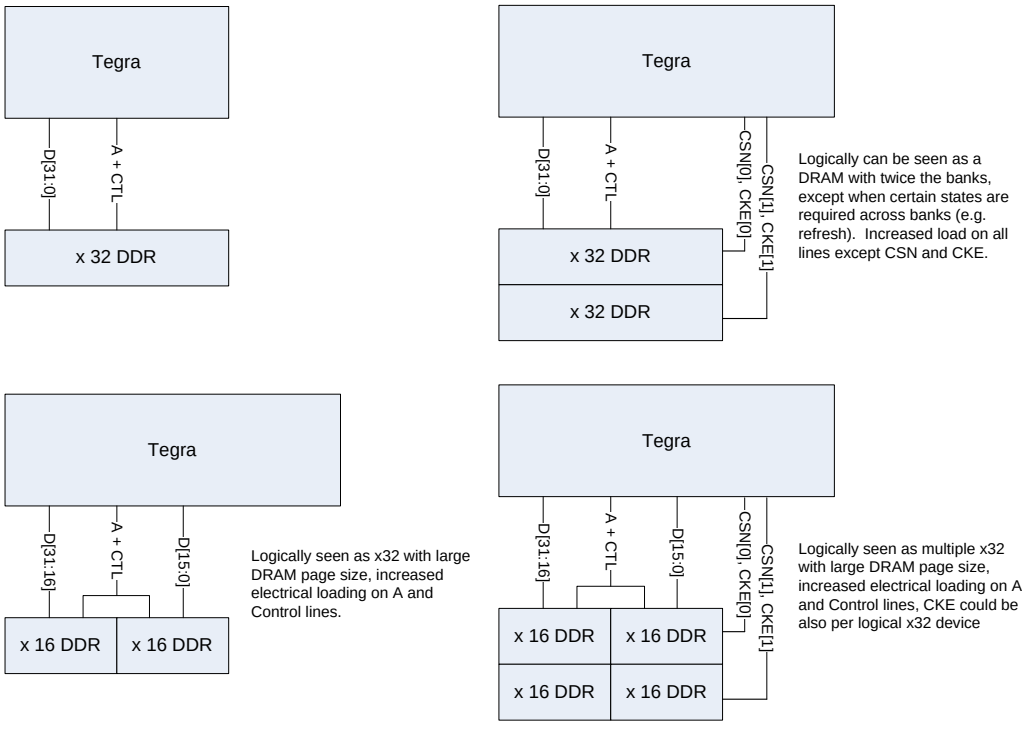
In addition to the TLB, page table updates require flushing stale entries out of the PTC. This needs to be done on a slot-by-slot basis by writing the PDE or PTE's physical address (PA[31:4]) to the PTC flush command. Each flush flushes four aligned PTEs or PDEs (one DRAM atom's worth) out of the cache.

A page table update requires three writes – one to write the PTE to memory, one to flush the stale PTE out of the PTC, and a third to flush the stale VA mapping out of the TLB. However, since a single PTC flush flushes four PTE or PDEs and a single TLB flush can flush the entire TLB, software can reduce the flush traffic by merging flushes from multiple updates.

20.8.4 DRAM Device, Bank and Page Mapping

This relates to the external memory devices, the EMC supports multiple devices with data widths of 8, 16, or 32 bits[7], all identical. Examples of configurations are shown in the next figure. The EMC controller uses specific bits in the address to identify the chip select associated with the address.

Figure 54: Multiple Chip Select Signals (x8 cases not shown)



The POR specifies support for 2 CS and 2 CKE. Having the same number of CS as CKE is important to limit the operating power consumption of the external DRAM.

[7] A x32 device may be a single x32 chip or two x16 chips in parallel.

The two ways of supporting multiple external devices have the following consequences for the inside logic.

- Width extension, when two x16 (or four x8) DRAM chips are used as one x32 device
  - Results in the apparent page size of the apparent x32 devices to be twice the size of the external x16 device page size but the number of banks is unchanged.
  - The EMC only needs one state machine corresponding to the apparent x32 device.
  - Data-bit loading is unchanged but address/control-bit loading is doubled.
- Depth extension, when multiple x32 devices are used at different address offsets,
  - Results in an apparent page size equal to the external device page size but the apparent number of banks is multiplied by the number of devices
  - The EMC needs to maintain multiple bank and device state machines
  - Loading on all interface bits except CKE and CS are doubled.

Each DRAM device also exposes a bank and page structure that affects performance. For better performance, a stream of accesses to the DRAM should be such that

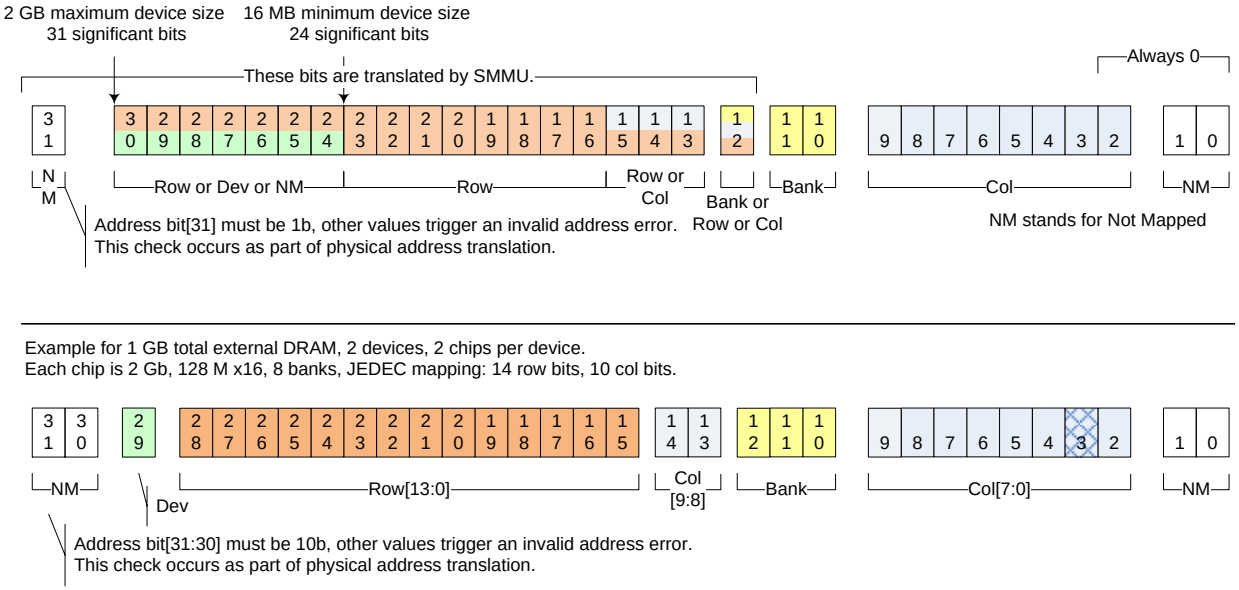
- Access to different banks are interleaved
- Consecutive accesses to the same bank are for the same page in that bank

The external memory arbiter attempts to generate a request stream with such characteristics, but it is dependent on the requests it receives from the different clients. It is more important to ensure bank interleaving for robust performance, this is achieved by ensuring that banks are also interleaved with a small pitch in the address space, with a pitch small compared to the size of expected data structures in memory. However, the pitch should not be too small either to ensure that address locality in a client results in same page accesses.

The proposed bank pitch corresponds to 64 DRAM data bursts, each of 16 bytes, for a final pitch of 1024 bytes. The bit mapping between the internal linear address and the device, row, column and bank bits is performed like this:

- The 2 LSBs of the linear address are ignored as the address granularity of the DRAM is 32 bits/4 bytes
- Bits[9:2] of the linear address are mapped as column bits[7:0]
- Bits [11:10] are bank bits.
- Bit [12] is a bank bit if the device has more than two bank bits.
- The next bits of the linear address are mapped as column bits, as many as remaining after previous mapping
- The next bits of the linear address are mapped as row bits, as many as needed for the selected device
- The next bit is a device bit if device bits are needed

Figure 55: Address Mapping Example



The number of bank, column, row, and device bits is limited by the number of address pins available:

- Bank width: 2 or 3
- Column width: 8 to 11
- Row width, depends on if DDR3 is being used with 1 device:
  - Yes: up to 16 bits
  - No: up to 15 bits
- Logical Devices (also known as chip selects): 1 or 2

When two logical devices are used, the total memory mapped by the second device must be less than or equal to the first device. The second device also may have a different row, bank, column mapping from the first device.

20.8.5 DRAM Power Modes

The following table maps between the system power modes and DRAM power modes.

Table 51: System Power Modes and DRAM Power Modes

	VDD_CORE	VDD_CPU	MC/EMC Power	PLL	MC/EMC Clock	DRAM State
LP0 (deep sleep)	off	off	off	off	off	self-refresh
LP1 (suspend)	on	off	on	off	off	self-refresh
LP2 (idle)	on	off	on	on	on	active

This subsection briefly describes the Tegra 4 system power modes. Refer to the PMC section for more details.

- LP0: The goal is ultra-long standby time in a cell phone. In this mode, major power rails are off and I/Os are held constant. DRAM is commanded by the AVP to enter self-refresh. The MC and EMC are powered off. Exiting LP0 is called warm boot, in comparison to cold boot which restarts the system. On warm boot, the MC and EMC are re-initialized from the PMC scratch registers and then DRAM is brought out of self-refresh. The LP0 entry code is executed from IRAM and the LP0 exit code is executed from boot ROM, both by the AVP.

- LP1: The goal is to suspend the CPU while allowing the hardware path to be externally accessible. All PLLs are turned off, and therefore MC/EMC/DRAM clocks are frozen. The DRAM is commanded by the AVP to enter self-refresh (it shares the same code with LP0 entry). Because the MC and EMC keep power during LP1, they do not need re-initialization. To exit LP1, simply enable the clock and exit self-refresh. Both LP1 entry and exit code are executed by the AVP from IRAM.
- LP2: The goal is to power off the CPU while still appearing functional to users. The DRAM can still be accessed by the display, audio, video, and other system modules, although no access comes from the CPU. The LP2 entry/exit sequences do not involve the MC/EMC/DRAM.

For pads and DRAM, LP0 and LP1 are in the same state, LP2 and normal running mode are in the same state.

20.8.6 Refresh and Self-Refresh

Two important EMC power features, DSR and DVFS, are related to refresh and self-refresh:

- For all DRAM types, a minimum number of refreshes are required in any rolling tREF window. A typical value is 4096 refreshes in a rolling 32ms window.
- If self-refreshes happen in a tREF rolling window, such time can be subtracted from the length of the tREF window, and the rest of the window requires the number of refreshes proportional to its length.
- DDR3 supports burst refreshes of up to 8 refreshes. LPDDR2/LPDDR3 does not set a maximum burst size, as long as the two requirements above are not violated.
- On self-refresh entry, the DRAM specification suggests that it is not necessary to issue a refresh in single refresh mode. In burst refresh mode, DDR3 does not require it, but LPDDR2 requires it.
  - LPDDR2
    - Burst refresh mode: recommended to issue a burst refresh (but may not be necessary)
    - Single refresh mode: not necessary to issue a refresh.
  - LPDDR3
    - LPDDR3 specification is not yet finalized, but Micron indicates that statements similar to LPDDR2 will be proposed.
  - DDR3
    - Both single and burst refresh: not necessary to issue refreshes (DDR3 supports burst refresh sizes of up to 8)
- For all DRAM types, refresh must be issued at self-refresh exit, because the last time of the internally scheduled self-refresh in DRAM is unknown.
- Beside the minimum refresh number requirement above, DRAM specifications also have the requirement on the maximum refresh number. On LPDDR2/LPDDR3, no more than 8 refreshes can be issued in any rolling tREFBW window. Without that parameter, DDR3 similarly requires no more than 16 refreshes issued in any 8 * tREFI rolling window. Since they are similar, the tREFBW function is reused for DDR3.

20.8.7 TrustZone Security Region

The MC can secure a region in the physical address aperture from insecure software sources. The region can be defined on 1MB boundaries in the external memory region, in multiples of 1 MB. If a non-secure request is made to the secured region, MC throws an interrupt and logs the details of the request (address, client ID). After the details are logged, write requests are dropped and read requests forced to return all 1's, thus protecting the secured region from corruption by the insecure source.

Only clients configured for security are able to make secure requests. In Tegra 4 devices, the CPU is configured to allow secure requests (via the PL310 and AXICIF), as well as the AHB slave clients and the VDE bitstream decode engine (BSEV). The AXICIF follows the AXI protocol of using aprot[1]==0 as the "secure" indication. The AHB bus adds a secure-bit as a

sideband to the request, and translates this bit to the MCCIF secure indication. The VDE BSEV connects directly to a secure MCCIF client.

The SMMU's memory client (PTCR, used for page-table misses) may also access secured memory if the SMMU is configured to allow secure accesses. See subsection on "Virtual Addressing" above for details.

In addition to securing the region in memory, register accesses that would modify the location of the secure region must also be secured. The two registers controlling security may only be modified by secure register requests.

20.8.8 OS-Protection Memory Region

The MC can secure a region in memory from access and modification by the GPU hardware engines. The intended use of this section is to prevent modification of the OS image from untrusted software via the hardware engines. The protected region is defined to start at EMEM_BOM and can be extended in multiples of 1MB. The clients allowed to access the protected region are the CPU, the COP, the AHB, VDE's BSEV, and the PTC and anything translated through the SMMU. All other clients attempting to access this region will cause MC to throw and interrupt and log the details of the request (address, client id). After the details are logged, write requests will be dropped and read requests will be forced to return all 1's, thus protecting the region from corruption by the insecure source.

In addition to protecting the region in memory, register accesses that would modify the boundary of the protected region are also secured by TrustZone.

20.9 Programming Model

There are three apertures that can be programmed, starting from EMC0_BASE, EMC1_BASE, and EMCB_BASE. The register reads/writes to EMC0/1_BASE go to a single EMC instance. The writes to EMCB_BASE go to both EMCs. The reads to EMCB_BASE go to EMC0TBD. Both EMC registers and IOBIST registers are accessed via the APB interface.

In single channel mode, EMC1 is not clocked or out of reset; therefore, register access to EMC1 might hang the APB bus.

There are three parts of software code access with MC/EMC registers: cold/warm boot (by either boot ROM or microboot), LP0/LP1 entry and LP1 exit, and DRAM DVFS sequence. It is also possible for another part of software to check the statistics registers of MC/EMC for testing, debugging, or performance monitoring.

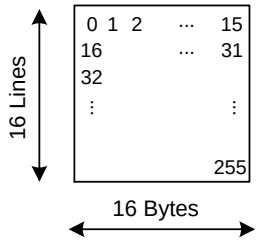
Cold boot and warm boot code are executed from the AVP, either off boot ROM or off microboot in IRAM. Warm boot is more sensitive to latency, so microboot is usually not a good option. Cold boot gets MC/EMC programming from the BCT table. Warm boot gets it from the PMC scratch registers, which contain a snapshot of MC/EMC register values stored by the Boot Loader, which is loaded to DRAM immediately after cold boot.

The DVFS sequence is in the OS kernel and run by the CPU off DRAM, so it is critical that the MC/EMC does not block traffic permanently at any time during the programming of this sequence.

20.10 Memory Tiling

The memory tiling format was originally proposed to decrease the latency for vertical accesses and to improve overall bandwidth utilization. Each tile is 256 bytes; the width of a tile is 16 bytes and height is 16 lines. Each tile entirely fits in a single DRAM page, often more than one per DRAM page. For example, 4 tiles will fit in a 1KB page. The addressing with in a tile is shown below and aligns with a macro-block pattern. This tiling fits well for video decoder and encoder needs. For clients that are inherently linear, tiling should have no effect for DRAM interfaces working on a 16-byte atom.

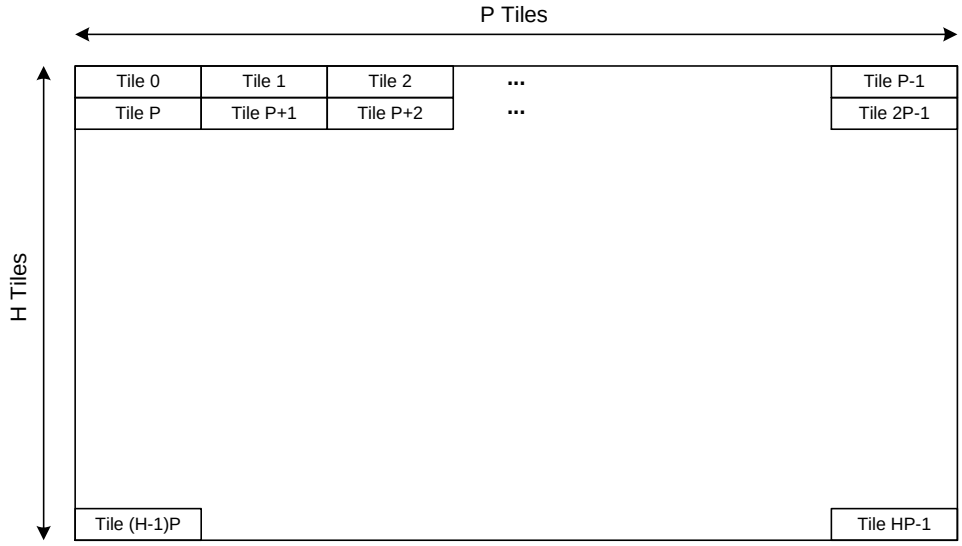
Figure 56: A Memory Tile



20.10.1 Tiling Pattern and Layout in Memory

Memory tiles are scan line ordered within tiled surface. Since each tile width is 16 bytes, height is 16 lines and the surface needs to be rectangular with integral number of tiles, it puts a restriction on the surface stride to be multiple of 16 bytes and height to be multiple of 16 lines. The diagram below shows how a typical surface of (PxH) tiles look like.

Figure 57: Tile Layout in a Surface



20.10.1.1 Tiled Address Calculation

For addressing a location in a tiled surface, the client has to provide the surface parameters - base address, pitch and pixel coordinate (x, y). The tiled address is calculated using the equation,

tiled_addr = base + (y/16) * 16 * pitch + (x/16) * 256 + (y%16) * 16 + x%16

where *base* is the surface base address in bytes, *y* is the vertical top-to-bottom offset of the pixel data in lines, *x* is the horizontal left-to-right offset of the pixel data in bytes, *pitch* is the surface padded pitch in bytes, and *tiled_addr* is the memory byte address after tiling.

20.10.1.2 Linear to Tiled Address Translation

Certain clients in the system are not able to provide (x, y) coordinates without disruptive hardware changes. In these cases, the client may choose to implement a linear-to-tiled address conversion mechanism that has reduced flexibility. By restricting the pitch *p* to the form

$p = k * 2^{(n + 6)}$

where *k* is an odd natural number and *n* is a natural number, a linear address can be converted to a tiled address requiring only the additional information of the surface base address and the surface pitch. This method reduces the width of the division operator required, thus reducing the latency impact of this operation.

The supported values of k are [1,3,5,7,9,11,13,15]; the supported values of n are [0,1,2,3,4,5,6]. For most surfaces with pitches of the restricted form, the resulting tiled address is exactly the same as would be computed by the normal (x, y) equation; surfaces where

$$(\text{linear} - \text{base}) > 2^{22}$$

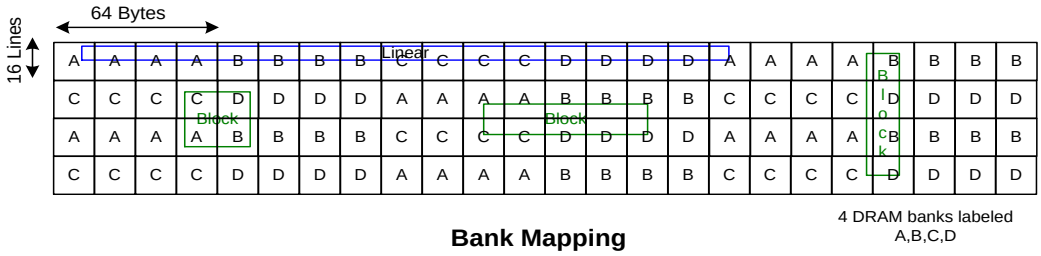
will result in incorrect tiled address. Certain surfaces that do not naturally align to a pitch of the form mentioned above will have wasted memory at the right and bottom edges of the surface.

20.10.2 Tiling Bank Layout

Even though tiling helps the block accesses to be in a single DRAM page, block accesses often fall in multiple tiles. For a typical DRAM, the penalty for closing the current page and opening a new page on the same bank is around 10 cycles; if the controller does not have to close the current page, this penalty drops to around 3 cycles. Penalty for closing and re-opening a page cannot be hidden in a single-client stream, so adjacent tiles should not be mapped to different row on the same bank. Penalty for opening a page on different bank can be hidden with the data transfer from different bank.

To balance the needs of linear clients and block clients, the bank interleaving must be chosen such that page-miss penalties are acceptable for both access patterns. The solution in Tegra 4 devices is to use byte address bits [11:10] as the bank bits to DRAM. For a typical 1KB-page DRAM, and assuming a 16-byte memory atom, this causes linear clients to switch banks every fourth access. The block clients also retain the benefits of page locality while still spreading their access patterns across enough banks to hide the page-open penalty.

Figure 58: Effects of Bank Address Mapping on Tile Layout



Bank Mapping

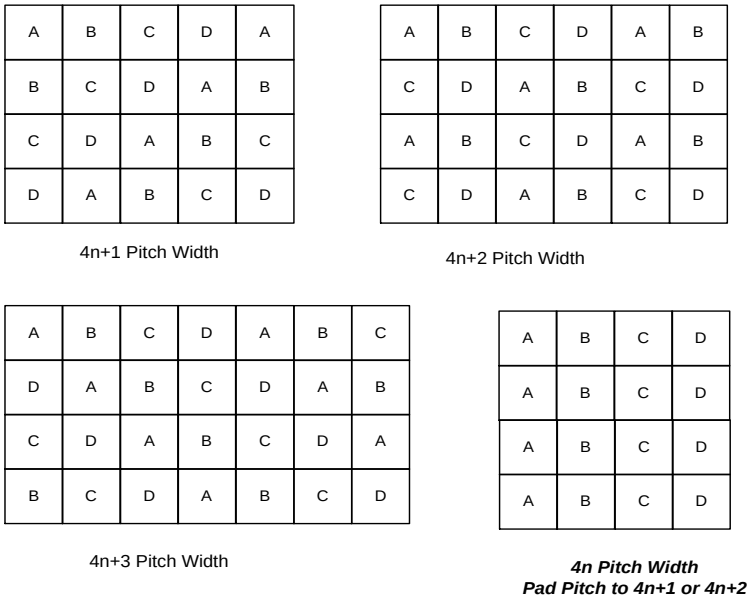
This bank mapping creates one problem, in that it reduces the inherent bank interleaving between different clients. The reordering aspects of the external memory arbiter should fairly handle contention between different clients; however the basic resource conflict still limits the available performance of the system.

20.10.2.1 Tiling Pitch Values and Padding

The figure below shows the bank ordering for different widths of surface. For pitch values which are in the form 4n+1, 4n+2, 4n+3, normal bank ordering provides bank interleaving for the vertical accesses. For 4n pitch, normal bank ordering is inefficient. To correct this, software should pad 4n pitch to 4n+1 or 4n+2 for better bank mapping for vertical accesses.

For 4n+1, 4n+3 mappings, even though banks in horizontal and vertical directions are different, the diagonals show inefficient bank orderings. Block accesses opening the same bank twice for these configurations will have some performance degradation. If this is not acceptable, the surfaces should be padded to a 4n+2 format.

Figure 59: Surface Pitch and Padding Examples



Impact of DRAM Configurations on Surface Pitch Alignment

Having 8 physical banks with the third bank-bit mapped low doubles the interval before a page-collision occurs (e.g., ABCDEFGH pattern) when using physical addressing. However, most applications are expected to use the virtual addressing via the SMMU, which translates the third bank-bit. Software SMMU allocation should be aware of this issue and attempt to page-color and pad accordingly.

Having two devices does not have any impact on tiling since the device bit is mapped high, and is translated.

Padding for Multiple Surfaces

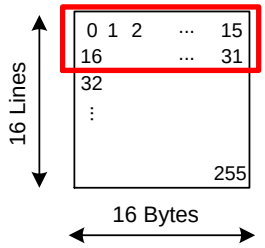
It is often the case that fetching from two surfaces which are tiled and switching buffers constantly. Good examples could be color and Z fetches, blending two surfaces, YUV surfaces, etc. In these cases it is quite possible that both the surfaces would keep accessing same bank and different rows, causing page-thrashing. This would create penalty in opening and closing the same bank multiple times. It must be noted that due to different consumption rates for different scenarios, it is impossible to avoid all page-thrashing. Efforts have been made to minimize page-thrashing in the MCCIF clients.

To reduce page-thrashing, software should program the base addresses of the surfaces staggered by the page size (usually 1KB), so that surfaces would start with a different bank. If the first surface is starting at address aligned to (4x) K bytes, the second surface should be aligned to (4x+1)K, (4x+2)K, or (4x+3)K.

20.10.3 Tiling with DRAM Atoms Larger than MSS Atoms

For situations where the DRAM fetches a minimum of 32 bytes per CAS command (DDR3), this tiling format produces an inherent 50% overfetch. Each memory access returns two lines of data, as shown in the figure below. For clients that issue request in macroblock order (MPE), or are natively working in 32 bytes (CPU caches, 3D engine) the MSS can coalesce two requests together late in the pipestage to take advantage of the overfetch. For clients that issue requests in horizontal linear order (Display, VI), the overfetch issue cannot be solved by MSS. Software should consider the overfetch inherent in linear clients into consideration when computing the overall bandwidth consumed for those clients touching tiled surfaces.

Figure 60: DDR3 Tiled Fetch



20.11 Memory Controller Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

20.11.1 MC Registers

USAGE NOTE: Many MC register fields are shadowed.

Writes to shadowed register fields update the shadow copy (this is default, assumes TIMING_CONTROL_DBG.WRITE_MUX==ASSEMBLY).

Reads to shadowed register fields return the currently-active copy (this is default, assumes TIMING_CONTROL_DBG.READ_MUX==ACTIVE).

This allows a new set of frequency-dependent timing parameters to be written to the shadow registers while memory traffic is ongoing, then when the parameters are completely written, the MC hardware can perform a timing-safe switch.

Such switches can be triggered via two methods: TIMING_CONTROL.TIMING_UPDATE (generally used during initialization), or the automatic CAR/EMC handshake on a clock-frequency or divider change (assuming TIMING_CONTROL_DBG.IGNORE_EMC_TRIGGERS==DISABLED).

Registers that are shadowed are marked by a comment:

> “This register is shadowed: see usage note at top”

USAGE NOTE: Many MC registers should be programmed by the Boot Loader as part of the warm boot (also known as "wake from LP0") and cold boot (also known as "power up") sequences. Suggested actions for the Boot ROM/Boot Loader are noted in comments labeled "Boot requirements: ...".

USAGE NOTE: Some MC registers may only be accessed by TrustZone-secured register transactions. See the Cortex-A15 documentation for information on how to handle security. Such registers are marked by a comment:

> “Access to this register is restricted to TrustZone-secured requestors.”

USAGE NOTE: The MC register fields to be stored in PMC scratch registers for warm boot must have "[PMC]" in their comments. After adding and removing such PMC flag, the tool warmboot_code_gen must be rerun to generate new boot ROM code.

20.11.1.1 MC_SMMU_CONFIG_0

Used for interrupt set/clear enums. When disabled, all transactions are untranslated. When enabled, transactions may be translated. See per-client and per-module enables in SMMU_* registers below.

This register can only be accessed by TrustZone-Secured accesses from the CPU.

Access to this register is restricted to TrustZone-secured requestors.

Boot requirements: This register should be saved to SDRAM registers and restored by the OS during warm boot.

SMMU Enable Register

Offset: 0x10 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
0	DISABLE	SMMU_ENABLE: 0 = DISABLE 1 = ENABLE

20.11.1.2 MC_SMMU_TLB_CONFIG_0

Translation Lookaside Buffer Config Register

Controls usage of the TLB.

Boot requirements: This register should be saved to SDRAM registers and restored by the OS during warm boot.

Offset: 0x14 | Read/Write: R/W | Reset: 0b001xxxxxxxxxxxxxxxxxxxxxxxx10000

Bit	Reset	Description
31	DISABLE	TLB_STATS_ENABLE: Enable TLB Hit and Miss counters 0 = DISABLE 1 = ENABLE
30	DISABLE	TLB_STATS_TEST: Set stats registers to all "1s" 0 = DISABLE 1 = ENABLE
29	ENABLE	TLB_HIT_UNDER_MISS: Allow hits to pass misses in the TLB. This value may not be changed on the fly. Ideally, this should be set before enabling the SMMU. At the very least, the TLB needs to be flushed and traffic through the SMMU stopped before changing this value. 0 = DISABLE 1 = ENABLE
4:0	0x10	TLB_ACTIVE_LINES: Set the number of active lines. Allows the TLB to be made "virtually smaller" to save power. "Inactive" lines will never hit and never hold data.

20.11.1.3 MC_SMMU_PTC_CONFIG_0

Controls usage of the PTC

Boot requirements: This register should be saved to SDRAM registers and restored by the OS during warm boot.

Page Table Cache Config Register

Offset: 0x18 | Read/Write: R/W | Reset: 0b001xxxxxxxxxxxxxxxxxxxxxxx111111

Bit	Reset	Description
31	DISABLE	PTC_STATS_ENABLE: Enable PTC Hit and Miss counters 0 = DISABLE 1 = ENABLE
30	DISABLE	PTC_STATS_TEST: Set stats registers to all "1s" 0 = DISABLE 1 = ENABLE
29	ENABLE	PTC_CACHE_ENABLE: Enable the PTC cache. 0 = DISABLE 1 = ENABLE
5:0	0x3f	PTC_INDEX_MAP: XOR pattern for tag generation

20.11.1.4 MC_SMMU_PTB_ASID_0

Page Table Bases (PTBs)

These pointers point to a 4KB-page aligned table of PDEs for each ASID. They also contain initial permission bits that are ANDed with those in the PDE and PTEs to determine the final permissions of accesses through this ASID.

The PTBs are stored in a RAM that is accessed indirectly though the SMMU_PTB_ASID and SMMU_PTB_DATA registers.

Boot requirements: This RAM should be saved to SDRAM and restored by the OS during warm boot.

Page Table Base ASID Register

This register selects which PTB is modified by the SMMU_PTB_DATA register.

Offset: 0x1c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
1:0	X	CURRENT_ASID: ASID used to address SMMU_PTB register

20.11.1.5 MC_SMMU_PTB_DATA_0

Writes or reads to this register write or read the Page Table Base pointer for the ASID defined in the SMMU_PTB_ASID register.

This register is not TrustZone secure, but only a TrustZone secure access may alter the PTB for a secure ASID.

Secure ASIDs have additional privilege over other non-secure ASIDs - they may fetch page tables from secure memory and they can allow clients translated through them to make secure accesses if programmed to do so.

Attempts to write a secure ASID with a non-secure access will be ignored, and non-secure reads will return garbage.

Page Table Base Data Register

Offset: 0x20 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	ASID_READABLE: if set, reads are allowed for this ASID
30	X	ASID_WRITABLE: if set, writes are allowed for this ASID
29	X	ASID_NONSECURE: if set, non-secure accesses are allowed for this ASID
19:0	X	ASID_PDE_BASE: Pointer to page of PDEs, bits [31:12] for this ASID

20.11.1.6 MC_SMMU_TLB_FLUSH_0

Software can write this register to flush a specific page group, 4MB section, entire ASID or the entire TLB. There is independent control over matching part or all of the VA and/or matching the ASID.

Note: The granularity of VA mapping here is at a page group, which is the number of PTEs that fit in a memory atom.

Translation Lookaside Buffer Flush Register

Offset: 0x30 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	TLB_FLUSH_ASID_MATCH: If enabled, only entries matching TLB_FLUSH_ASID are flushed. 0 = DISABLE 1 = ENABLE
30:29	X	TLB_FLUSH_ASID: ASID to match when TLB_FLUSH_ASID_MATCH is ENABLED
19:2	X	TLB_FLUSH_VA_GROUP: Virtual address to match for group flushes, VA[31:14]
1:0	X	TLB_FLUSH_VA_MATCH: Controls flushing based on VA, one of ALL: Flush entire TLB SECTION: Flush all entries with a VA[31:22] that match TLB_FLUSH_VA_SECTION GROUP: Flush all entries with a VA[31:14] that match TLB_FLUSH_VA_GROUP 0 = ALL 2 = SECTION 3 = GROUP

20.11.1.7 MC_SMMU_PTC_FLUSH_0

Software can use this register to flush a specific PTE or PDE from the Page Table Cache (PTC) by writing the atom-aligned physical address of the PTE group. It can also flush the entire PTC via this register.

Note: The granularity of flushing is at the PTE group, which is the number of PTEs that fit in a memory atom.

Page Table Cache Flush Register

Offset: 0x34 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:4	X	PTC_FLUSH_ADR: Physical address of PTE group to match for address flushes, PA[31:4]
0	X	PTC_FLUSH_TYPE: Controls flushing, one of ALL: Flush entire PTC ADR: Flush PTEs that are addressed by PTC_FLUSH_ADR 0 = ALL 1 = ADR

20.11.1.8 MC_SMMU_ASID_SECURITY_0

This register controls which ASIDs are considered "secure". Secure ASIDs cannot have their PTBs read or written by a non-secure access, nor can they be assigned as a module's ASID by a non-secure write.

ASIDs may also be enabled as "promoting", which allows modules mapped through them to inherit their secure status. If a secure ASID has its promotion bit enabled, modules mapped to this ASID may access the secure physical memory region.

> **Note:** All ASIDs, regardless of security, may access TrustZone-secure physical memory for page table accesses.

This register can only be accessed by TrustZone-Secured accesses from the CPU.

Boot requirements: This register should be saved to SDRAM and restored by the OS during warm boot.

ASID Security Register

Offset: 0x38 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0bxxxxxxxxxxxx0000xxxxxxxxxxxx0000

Bit	Reset	Description
19:16	0x0	PROMOTING_ASIDS: If bit N in this field is set, clients that are mapped to ASID N may access the TrustZone-secure area of memory if ASID N is also marked as "secure".
3:0	0x0	SECURE_ASIDS: If bit N in this field is set, the PTB of ASID N cannot be read or written by a non-secure access, nor can this ASID be mapped to a module by a non-secure write.

20.11.1.9 MC_EMEM_CFG_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.
- During warm boot, this register may be derived from EMEM_ADR_CFG* register values.

External Memory Aperture Configuration

Offset: 0x50 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxx0011111111111

Bit	Reset	Description
11:0	0x7ff	EMEM_SIZE_MB: Specify the external memory aperture size in megabytes. This field must be set to less than or equal to the available physical memory for proper operation. This value is used to check for decode errors; requests to addresses greater than or equal to EMEM_BOM+EMEM_SIZE_MB will result in an EMEM decode error.

20.11.1.10 MC_EMEM_ADR_CFG_0

The EMEM_ADR_CFG* registers are used to specify the DRAM density and geometry parameters. MC will use these parameters to derive device, row, bank, column addressing values to EMC.

> **Note:** The maximum size externally supported is dependent on pinout and address-map aperture limitations, and that it may be possible to configure the device/row/bank/column addresses in ways that exceed these limitations.

For each device attached [1..NUMDEV], the associated per-device address configuration must also be programmed, in the associated register EMEM_ADR_CFG_DEV{N-1}.

The maximum number of row bits carried internally is defined by NV_MC_EMEM_ROW_WIDTH.

The maximum number of address pins is defined by NV_MC_EMEM_ROWPIN_WIDTH.

If not being used to address a device, the chip-select pins may be re-used as row address pins for DDR3.

> **Note:** LPDDR2 only supports up to 15 row bits.

Valid settings for EMEM_ADR_CFG_DEV* registers should ensure device address width <= bank width + column width + NV_MC_EMEM_ROW_WIDTH

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

External Memory Address Configuration, System

Offset: 0x54 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
0	N1	EMEM_NUMDEV: [PMC] number of populated DRAM devices. 0 = N1 1 = N2

20.11.1.11 MC_EMEM_ADR_CFG_DEV0_0

Configures the density and geometry of the first attached device.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

External Memory Address Configuration, Device 0

Offset: 0x58 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxx0100xxxxxx10xxxxx010

Bit	Reset	Description
19:16	D64MB	EMEM_DEV0_DEVSIZE: [PMC] density of the first attached DRAM device, used to generate width of row address. 2 = D16MB 3 = D32MB 4 = D64MB 5 = D128MB 6 = D256MB 7 = D512MB 8 = D1024MB 8 = D1GB 9 = D2048MB 9 = D2GB
9:8	W2	EMEM_DEV0_BANKWIDTH: [PMC] width of bank address of the first attached DRAM device. 2 = W2 3 = W3
2:0	W9	EMEM_DEV0_COLWIDTH: [PMC] width of column address of the first attached DRAM device. W8 maps to 1KB page size, W9 to 2KB, W10 to 4KB, etc. 1 = W8 2 = W9 3 = W10 4 = W11

20.11.1.12 MC_EMEM_ADR_CFG_DEV1_0

Configures the density and geometry of the second attached device, if a second device is populated.

To allow for 2 devices with different colwidth/bankwidth to be used without creating holes in the system-level address map, the DEVSIZE setting of the second device may be smaller than the others to allow for non-powers-of-2 attached memory sizes.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

External Memory Address Configuration, Device 1

Offset: 0x5c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxx0100xxxxxx10xxxxx010

Bit	Reset	Description
19:16	D64MB	EMEM_DEV1_DEVSIZE: [PMC] density of the second attached DRAM device, used to generate width of row address. 2 = D16MB 3 = D32MB 4 = D64MB 5 = D128MB 6 = D256MB 7 = D512MB 8 = D1024MB 8 = D1GB 9 = D2048MB 9 = D2GB
9:8	W2	EMEM_DEV1_BANKWIDTH: [PMC] width of bank address of the second attached DRAM device. 2 = W2 3 = W3
2:0	W9	EMEM_DEV1_COLWIDTH: [PMC] width of column address of the second attached DRAM device. W8 maps to 1KB page size, W9 to 2KB, W10 to 4KB, etc. 1 = W8 2 = W9 3 = W10 4 = W11

20.11.1.13 MC_SECURITY_CFG0_0

Security aperture: can be only accessed by TrustZone-Secured accesses from the secure clients. Access to this register is restricted to TrustZone-secured requestors.

Boot requirements: This security configuration register should be saved to SDRAM and restored by the OS during warm boot.

Generated from mc_clientsecurity.spec

The global memory client IDs with TrustZone-level security are as follows:

CLIENT	ID	ID(hex)	SWGROUP	MODULE	PARTITION	DESCRIPTION
csr_ppcsahbslvr	30	(0x1e)	ppcs	ppcs	ahb	AHB bus reads
csr_vdebsevr	34	(0x22)	vde	vde2x	vd	Video Decode (VDE) video bitstream engine reads
csr_mpcorelpr	38	(0x26)	mpcorelp	mpcorelp	a15lp	Reads from Cortex-A15 Shadow CPU core via the L2 cache
csr_mpcorer	39	(0x27)	mpcore	mpcore	pi	Reads from 4 Cortex-A15 CPU cores via the L2 cache

CLIENT	ID	ID(hex)	SWGROUP	MODULE	PARTITION	DESCRIPTION
csw_mpcorelpw	56	(0x38)	mpcorelp	mpcorelp	a15lp	Writes from Cortex-A15 Shadow CPU core via the L2 cache
csw_mpcorew	57	(0x39)	mpcore	mpcore	pi	Writes from 4 Cortex-A15 CPU cores via the L2 cache
csw_ppcsahbslvw	60	(0x3c)	ppcs	ppcs	ahb	AHB bus writes
csw_vdebsevw	62	(0x3e)	vde	vde2x	vd	Video Decode (VDE) video bitstream engine writes

Secure Region Configuration: Base

Offset: 0x70 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0b100000000000xxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:20	0x800	SECURITY_BOM: SECURITY_BOM is the base of the secured region, limited to MB granularity. This must point to a region of the physical address map allocated to EMEM for it to be effective; MC cannot secure address space it does not own. (In other words, this is an absolute address, not an offset.) Above is the list of clients with the TrustZone-security access. Note that AXICIF clients will adhere to the standard AXI protocol "aprot[1]==0" indication for secure requests.

20.11.1.14 MC_SECURITY_CFG1_0

Access to this register is restricted to TrustZone-secured requestors.

Boot requirements:

This security configuration register should be saved to SDRAM and restored by the OS during warm boot.

Secure Region Configuration: Bound

Offset: 0x74 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0bxxxxxxxxxxxxxxxxxxxx000000000000

Bit	Reset	Description
11:0	0x0	SECURITY_SIZE_MB: SECURITY_SIZE_MB is the size, in megabytes, of the secured region. If set to 0, the security check in MC is disabled.

20.11.1.15 MC_SECURITY_CFG2_0

Protected/Carveout apertures: If enabled, certain memory clients are restricted to using SMMU-translated pages or the carveout aperture, they cannot access the protected aperture.

The protected region can be used to prevent hardware engines from accessing the operating system.

The protected region is always [EMEM_BOM..CARVEOUT_BOM).

The carveout region is always [CARVEOUT_BOM..EMEM_TOM].

Access to this register is restricted to TrustZone-secured requestors.

Boot requirements:

This security configuration register should be saved to SDRAM and restored by the OS during warm boot.

The global memory client IDs allowed access to the protected region are as follows:

CLIENT	ID	ID(hex)	SWGROUP	MODULE	PARTITION	DESCRIPTION
csr_ptcr	0	(0x00)	ptc			Misses from System Memory Management Unit (SMMU) Page Table Cache (PTC)
csr_avpcarm7r	15	(0x0f)	avpc	avpc	avp	ARM7 Audio-Video Processor (AVP) reads via the avp_cache
csr_ppcsahbslvr	30	(0x1e)	ppcs	ppcs	ahb	AHB bus reads
csr_vdebsevr	34	(0x22)	vde	vde2x	vd	Video Decode (VDE) video bitstream engine reads
csr_mpcorelpr	38	(0x26)	mpcorelp	mpcorelp	a15lp	Reads from Cortex-A15 Shadow CPU core via the L2 cache
csr_mpcorer	39	(0x27)	mpcore	mpcore	pi	Reads from 4 Cortex-A15 CPU cores via the L2 cache
csw_avpcarm7w	50	(0x32)	avpc	avpc	avp	ARM7 Audio-Video Processor (AVP) writes via the avp_cache
csw_mpcorelpw	56	(0x38)	mpcorelp	mpcorelp	a15lp	Writes from Cortex-A15 Shadow CPU core via the L2 cache
csw_mpcorew	57	(0x39)	mpcore	mpcore	pi	Writes from 4 Cortex-A15 CPU cores via the L2 cache
csw_ppcsahbslvw	60	(0x3c)	ppcs	ppcs	ahb	AHB bus writes
csw_vdebsevw	62	(0x3e)	vde	vde2x	vd	Video Decode (VDE) video bitstream engine writes

Protected Region Configuration: Bound

Offset: 0x78 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0b100000000000

Bit	Reset	Description
31:20	0x800	CARVEOUT_BOM: CARVEOUT_BOM is the megabyte-aligned starting address of the carveout region. (In other words, this is an absolute address, not an offset.) If CARVEOUT_BOM is set to <=NV_ADDRESS_MAP_EMEM_BASE, this security check in MC is disabled.

20.11.1.16 MC_EMEM_ARB_CFG_0

General configuration of the External Memory Arbiter.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM warm boot, this register may be derived from the emcclk and a standard tick-value.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration

Offset: 0x90 | Read/Write: R/W | Reset: 0b00000xxxxxxx000001000

Bit	Reset	Description
20:16	0x0	EXTRA_TICKS_PER_UPDATE: The number of extra ticks to add every deadline timer update. Used to keep cycles-per-tick constant when the mcclk period is less than the chosen wall-clock granularity for the deadline counter. If =0, then every tick increments the deadline counter by 1; if =1, then every tick increments the deadline counter by 2, etc. Assuming a target of 30ns-per-tick, some example programming (assumes DIV=2): 10MHz emcclk => 5MHz mcclk => would need to increment deadline counter by 6.67 if using 1 cycle-per-tick => or update by 20 every 3 cycles => 3 cycles-per-tick and 19 extra-ticks-per-update. Since performance is not critical at such slow speeds, rounding errors may be deemed acceptable, so the previous example may be simpler to program as 1 cycle-per-tick and 5 extra-ticks-per-update. Another example, the slowest supported clock speed using a 30ns tick that doesn't distort the tick is: 1.04MHz mcclk => with 31 extra-ticks-per-update and updating every cycle.
8:0	0x8	CYCLES_PER_UPDATE: The number of mcclk cycles per deadline timer update. The target wall-clock granularity ("tick") for the deadline counter should be fixed for the design, then the CYCLES_PER_UPDATE should be used to convert the wall-time value into mcclk cycles. All client latency allowances are also expressed in units of "ticks". Of all deadline-related controls, only the CYCLES_PER_UPDATE and EXTRA_TICKS_PER_UPDATE values need to be updated on a clock-change event. Assuming a target of 30ns-per-tick, some example programming (assumes DIV=2): 667MHz emcclk => 333MHz mcclk => 10 cycles-per-tick; 400MHz emcclk => 200MHz mcclk => 6 cycles-per-tick. Note that value 0x0 is illegal.

20.11.1.17 MC_EMEM_ARB_OUTSTANDING_REQ_0

This register can be used to limit the number of outstanding requests in the arbiter and monitor the count.

If outstanding transaction is greater than the maximum, requests are throttled based on MC_EMEM_ARB_RING1_THROTTLE_0 register

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from other MC settings.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: Outstanding Request Limiter

Offset: 0x94 | Read/Write: R/W | Reset: 0b10xxxxxxxxxxxxxxxxxxxxx010000000

Bit	R/W	Reset	Description
31	RW	ENABLE	LIMIT_OUTSTANDING: when ENABLED, the total number of requests in the arbiter is limited to the value in ARB_MAX_OUTSTANDING 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	LIMIT_DURING_HOLDOFF_OVERRIDE: when DISABLED, override the limiting of transactions during hold-off to let requests flow freely 0 = DISABLE 1 = ENABLE
24:16	RO	X	ARB_OUTSTANDING_COUNT: Current number of outstanding requests
8:0	RW	0x80	ARB_MAX_OUTSTANDING: The maximum number of requests allowed in the arbiter when LIMIT_OUTSTANDING is set to ENABLE.

20.11.1.18 MC_EMEM_ARB_TIMING_RCD_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: DRAM Timing : tRCD

Offset: 0x98 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111

Bit	Reset	Description
5:0	0x3f	RCD: The minimum number of cycles between activate commands to the same bank. Program to max (1,ceil(max(EMC.WR_RCD,EMC.RD_RCD)/DIV)-1-1). Note that value 0x0 is illegal.

20.11.1.19 MC_EMEM_ARB_TIMING_RP_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: DRAM Timing : tRP

Offset: 0x9c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111

Bit	Reset	Description
6:0	0x7f	RP: The minimum number of cycles between an internal precharge command and an activate command to the same bank. Program to ceil (EMC.RP/DIV)-1+SFA. The combined value of RAP2PRE+RP 0x0 is illegal; the combined value of WAP2PRE+RP 0x0 is illegal.

20.11.1.20 MC_EMEM_ARB_TIMING_RC_0

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: DRAM Timing : tRC

Offset: 0xa0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111

Bit	Reset	Description
6:0	0x7f	RC: This is the minimum number of cycles between activate commands to the same bank. Program to ceil (max(EMC.RC,(EMC.RAS+EMC.RP))/DIV)-1.

20.11.1.21 MC_EMEM_ARB_TIMING_RAS_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: DRAM Timing: tRAS

Offset: 0xa4 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111

Bit	Reset	Description
5:0	0x3f	RAS: This is the minimum number of cycles between an activate command and a precharge command to the same bank. Program to ceil (EMC.RAS/DIV)-1-1.

20.11.1.22 MC_EMEM_ARB_TIMING_FAW_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: DRAM Timing: tFAW

Offset: 0xa8 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx1111111

Bit	Reset	Description
5:0	0x3f	FAW: For 8-bank devices: Only 4 activates may occur within this rolling window. Program to ceil (EMC.TFAW/DIV)-1. Programming this to 0x0 or not programming a device to 4 banks will turn off this check.

20.11.1.23 MC_EMEM_ARB_TIMING_RRD_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at top of spec file.

External Memory Arbitration Configuration: DRAM Timing: tRRD

Offset: 0xac | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111

Bit	Reset	Description
3:0	0xf	RRD: The minimum number of cycles between an activate command and an activate command to a different bank. Program to ceil (EMC.RRD/DIV)-1.

20.11.1.24 MC_EMEM_ARB_TIMING_RAP2PRE_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: DRAM Timing : tRAP2PRE

Offset: 0xb0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111

Bit	Reset	Description
5:0	0x3f	RAP2PRE: The number of cycles between a read command with auto-precharge and the internal precharge command generated by the DRAM. Program to ceil (EMC_R2P/DIV). Note that: the combined value of RAP2PRE+RP 0x0 is illegal.

20.11.1.25 MC_EMEM_ARB_TIMING_WAP2PRE_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: DRAM Timing: tWAP2PRE

Offset: 0xb4 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111

Bit	Reset	Description
6:0	0x7f	WAP2PRE: The number of cycles between a write command with auto-precharge and the internal precharge command generated by the DRAM. Program to ceil (EMC_W2P/DIV). Note that: the combined value of WAP2PRE+RP 0x0 is illegal.

20.11.1.26 MC_EMEM_ARB_TIMING_R2R_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: DRAM Timing: tR2R

Offset: 0xb8 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111

Bit	Reset	Description
4:0	0x1f	R2R: The number of cycles between consecutive read commands to different devices (different chip selects). Program to ceil (EMC.REXT/DIV)-1+OTFA+SFA.

20.11.1.27 MC_EMEM_ARB_TIMING_W2W_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: DRAM Timing: tW2W

Offset: 0xbc | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111

Bit	Reset	Description
4:0	0x1f	W2W: The number of cycles between consecutive write commands to different devices (different chip selects). Program to ceil (EMC.WEXT/DIV)-1+SFA.

20.11.1.28 MC_EMEM_ARB_TIMING_R2W_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: DRAM Timing: tR2W

Offset: 0xc0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111

Bit	Reset	Description
5:0	0x3f	R2W: The number of cycles to turn the bus from reads to writes. Program to ceil (EMC.R2W/DIV)-1+OTFA+SFA.

20.11.1.29 MC_EMEM_ARB_TIMING_W2R_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from EMC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: DRAM Timing: tW2R

Offset: 0xc4 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111

Bit	Reset	Description
5:0	0x3f	W2R: The number of cycles to turn the bus from reads to writes. Program to ceil (EMC.W2R/DIV)-1+SFA.

20.11.1.30 MC_EMEM_ARB_DA_TURNS_0

The direction arbiter attempts to choose the highest efficiency (i.e. lowest bubbles) direction to open pages next. While programming this register incorrectly may lower performance, it should never cause chip-hangs since the EMC always controls the final DRAM timing decisions. This register should be reprogrammed after clock-change events to bring the computed values back in alignment with EMC's register values. These turn costs are deliberately not computed in hardware to provide better flexibility to tweak arbiter performance.

The configuration registers for turns for DA are separate from the timing registers to allow for flexibility in the programming.

Boot requirements:

- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from MC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: Direction Arbiter: Turns

Offset: 0xd0 | Read/Write: R/W | Reset: 0b00001111000011110000001000000101

Bit	Reset	Description
31:24	0xf	W2R_TURN: Bubbles produced by a write to read bus turn. Approximately MC_EMEM_ARB_TIMING_W2R.
23:16	0xf	R2W_TURN: Bubbles produced by a read to write bus turn. Approximately MC_EMEM_ARB_TIMING_R2W.
15:8	0x2	W2W_TURN: Bubbles produced by a write to write (other device) bus turn. Approximately MC_EMEM_ARB_TIMING_W2W.
7:0	0x5	R2R_TURN: Bubbles produced by a read to read (other device) bus turn. Approximately MC_EMEM_ARB_TIMING_R2R.

20.11.1.31 MC_EMEM_ARB_DA_COVERS_0

The direction arbiter attempts to choose the highest efficiency (i.e., lowest bubbles) direction to open pages next. While programming this register incorrectly may lower performance, it should never cause chip-hangs since the EMC always controls the final DRAM timing decisions. This register should be reprogrammed after clock-change events to bring the computed values back in alignment with EMC's register values. These cover costs are deliberately not computed in hardware to provide better flexibility to tweak arbiter performance.

Boot requirements:

- This arbitration configuration register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from MC settings using the given equations.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: Direction Arbiter: Covers

Offset: 0xd4 | Read/Write: R/W | Reset: 0bxxxxxxxx00001111000001111000011111

Bit	Reset	Description
23:16	0xf	RCD_W_COVER: Cost to cover the activate-to-write delay. Approximately ceil ((EMC.RTP + EMC.RP + EMC.RD_RCD)/DIV)-1.
15:8	0xf	RCD_R_COVER: Cost to cover the activate-to-read delay. Approximately ceil ((EMC.WTP + EMC.RP + EMC.WR_RCD)/DIV)-1.

Bit	Reset	Description
7:0	0xf	RC_COVER: Cost to cover the activate-to-activate delay. Approximately MC_EMEM_ARB_TIMING_RC.

20.11.1.32 MC_EMEM_ARB_MISC0_0

BC2AA_HOLDOFF: In the case where the currently-open pages have more work pending than the time it would take to open another page, it is better to hold off opening that page until it is truly needed. The Best-bank Cache computes the pending work and compares it to this threshold and advises the Activation Arbiter to hold-off.

Priority Inversion: In general, if the arbiter is working on a lower-priority request that blocks a higher-priority request, these thresholds are used to limit the number of low-priority requests we will service before the arbiter will interrupt the lower-priority traffic to start servicing the higher-priority traffic.

The priority classes are (in increasing priority order): Low (! expired), High (expired), HighIso.

There are three ways priority can be inverted: Bank Activation, Bus Direction, and Refresh:

Bank Activation: If an unexpired request to bank A, row R currently holds the page open, and a request to bank A, row S expires, the Transaction Arbiter must make a decision whether to finish out the work for row R or to interrupt that transfer to immediately service row S. The TA makes this decision based on the remaining work for row R and whether row S is for an isochronous client, and the two PRIORITY_INVERSION thresholds.

Bus Direction: If the Transaction Arbiter is currently working on expired requests in direction P, and requests are also expired (or expired-iso) in direction Q, the Transaction Arbiter must make a decision whether to continue working in direction P or switch the bus to direction Q. The Transaction Arbiter will service at maximum PRIORITY_INVERSION_THRESHOLD (or _INVERSION_ISO_THRESHOLD) requests in direction P before switching to direction Q.

Refresh: If the Transaction Arbiter is currently working on requests, and EMC asks to inject a refresh, the Transaction Arbiter must decide whether it is more efficient (overall) to close the pages immediately or to finish the pages and then close them. Depending on the setting of the REFRESH_ACK_THRESHOLD_USAGE configuration bit (in EMEM_ARB_MISC1 register, below), the Transaction Arbiter will either choose to close the page immediately (NONE), close the page immediately if its length is over the iso threshold (ISO), or close the page immediately if its length is over the normal threshold (NORMAL). If the length is less than the chosen threshold, the Transaction Arbiter will service all remaining requests to the page before acknowledging the EMC refresh request.

When in DDR3 coalesce-for-performance mode, the arbiters will monitor the traffic patterns and may halve the priority inversion thresholds if they detect the current traffic is not coalescing efficiently. When in MC_EMC_SAME_FREQ mode, the priority inversion thresholds should be set to half of their non-same-frequency values.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- The MC_EMC_SAME_FREQ field should be saved to scratch registers and restored by the Boot ROM during warm boot.
- The other fields of this register should be saved to SDRAM and restored by the OS during warm boot.
- During Boot ROM section of warm boot, this register may be derived from other settings.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: Miscellaneous Thresholds (0)

Offset: 0xd8 | Read/Write: R/W | Reset: 0bx111001000001010x100000000010000

Bit	Reset	Description
30:28	0x7	ATOMS_PER_DVFS_PULSE: The MC will toggle the sys_stats_mon outputs for every 2^(atoms_per_dvfs_pulse+1) atoms of traffic, where an atom is defined as NV_MC_EMEM_WD=16 bytes. The maximum atoms_per_dvfs_pulse setting supported is the reset value, also known as NV_MC_EMEM_DVFS_CNTR_CFG_RESET.
27	DISABLE	MC_EMC_SAME_FREQ: [PMC] Used in conjunction with EMEM_ARB_MISC1.COALESCE_FOR_PERFORMANCE to configure how often MC can send EMC a data request, should be configured to mirror CAR's CLK_SOURCE_EMC.MC_EMC_SAME_FREQ. If configured to DIV=2 BL=4, then MC can send a data request every cycle. If configured to DIV=1 BL=4, then MC can send a data request every other cycle. If configured to DIV=2 BL!=4, then MC can send a data request every other cycle. If configured to DIV=1 BL!=4, then MC can send a data request once every four cycles. 0 = MC is 1/2 the frequency of EMC (DIV=2). 1 = MC is the same frequency of EMC (DIV=1). 0 = DISABLE 1 = ENABLE
26:21	0x10	EXPIRING_SOON_SLACK_THRESHOLD: Slack threshold below which an isochronous request is considered to be expiring "soon". Used to de-assert mc2emc_idle signal early to wake EMC out of power-down or self-refresh. If using EMC's Dynamic Self Refresh features, set this to the ticks needed to cover exit self-refresh delay; otherwise, if using EMC's ACPD feature, set to the ticks needed to cover exit power-down delay; if neither power-saving feature is being used, the programming of this threshold should have no effect.
20:16	0xa	PRIORITY_INVERSION_ISO_THRESHOLD: Bank Activation: maximum number of unexpired or expired-but-not-isochronous requests that can be serviced before switching to expired isochronous traffic. Bus Direction: maximum number of requests in direction P that can be serviced before switching to expired-isochronous traffic in direction Q. Refresh: maximum number of requests that can be serviced before refresh is sent. Values <= 0x2 disable the feature. Values < 0x6 should be avoided if coalescing for performance.
14:8	0x40	PRIORITY_INVERSION_THRESHOLD: Bank Activation: maximum number of unexpired requests that can be serviced before switching to expired traffic. Bus Direction: maximum number of requests in direction P that can be serviced before switching to expired traffic in direction Q. Refresh: maximum number of requests that can be serviced before refresh is sent. Values <= 0x2 disable the feature. Values < 0x6 should be avoided if coalescing for performance.
7:0	0x10	BC2AA_HOLDOFF_THRESHOLD: If the pending work is greater than this value, hold off on activations. Generally should be set to match the page-opening cost: MC_EMEM_ARB_TIMING_RC+1.

20.11.1.33 MC_EMEM_ARB_MISC1_0

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- The COALESCE_FOR_PERFORMANCE field should be saved to scratch registers and restored by the Boot ROM during warm boot. It may also be derived from EMC settings.
- The other fields of this register should be saved to SDRAM and restored by the OS during warm boot.

External Memory Arbitration Configuration: Miscellaneous Thresholds (1)

Offset: 0xdc | Read/Write: R/W | Reset: 0b10000x0010101xxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:28	0x8	DEADLOCK_PREVENTION_SLACK_THRESHOLD: If the slack at the head of any bank-queue, or any of the slacks in the hit-under-miss FIFO is less than -(2^threshold), backpressure the input to the arbiter until the violating slack values have been arbitrated. This is intended to: (a) avoid deadline-wrapping in heavily backpressured corner-cases and (b) quickly get such cases back into high-efficiency arbitration. Value of 0x0 disables the feature. The reset value is NV_MC_EMEM_DL_WIDTH-1. Behavior with values >= NV_MC_EMEM_DL_WIDTH is UNDEFINED. If the threshold is reached, ARBITRATION_EMEM_INT will fire.
27	DISABLE	COALESCE_FOR_PERFORMANCE: If EMC has BL=8 or BL=4_or_8_on_the_fly, tell the arbiter to coalesce for performance reasons. While MC/EMC will functionally work without this bit, performance will be significantly better if MC knows it should coalesce. 0 = DISABLE 1 = ENABLE
25:24	NORMAL	REFRESH_ACK_THRESHOLD_USAGE: Determines which thresholds (if any) the refresh-acknowledge signal will use to determine how to close all pages after EMC has requested a refresh. 0 = NORMAL : Use priority-inversion threshold 1 = ISO : Use priority-inversion-iso threshold 2 = NONE : Force a precharge immediately to close the page

20.11.1.34 MC_EMEM_ARB_RING1_THROTTLE_0

Ring1 Snap Arbiter Throttle - Inserts some number of stall cycles at Ring1 after every request.

Throttle cycle count varies whether outstanding_request count is below (LOW) or above (HIGH) the max_outstanding threshold.

Boot requirements: During Boot ROM section of warm boot, this register may be derived from other settings.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: Snap Arbiter Throttle

Offset: 0xe0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxx11111xxxxxxxxxxx00000

Bit	Reset	Description
20:16	0x1f	RING1_THROTTLE_CYCLES_HIGH: Cycles of throttle after each request when outstanding transaction count is greater than or equal to threshold.
4:0	0x0	RING1_THROTTLE_CYCLES_LOW: Cycles of throttle after each request when outstanding transaction count is below threshold. Suggested programming: (DIV==1)? 1 : 0.

20.11.1.35 MC_EMEM_ARB_RING3_THROTTLE_0

Ring3 Snap Arbiter Throttle - Inserts some number of stall cycles at Ring3 after every request

Boot requirements: This register should be saved in the SDRAM and restored by the OS during warm boot.

This register is shadowed: see usage note at top.

External Memory Arbitration Configuration: Snap Arbiter Throttle

Offset: 0xe4 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxx00000xxxxxxxxxxx00000

Bit	Reset	Description
4:0	0x0	RING3_THROTTLE_CYCLES: Cycles of throttle after each request.

20.11.1.36 MC_CLIENT_HOTRESET_CTRL_0

Writing FLUSH_ENABLE to a bit in this register causes a flush to be performed on all of the clients for the selected swname. The status of the flush (done/in progress) can be monitored in the CLIENT_HOTRESET_STATUS register.

A proper client reset sequence is as follows:

1. Set the appropriate FLUSH_ENABLE bit in the CLIENT_HOTRESET_CTRL register to block further access by the client, and start the flush process.
2. Poll the CLIENT_HOTRESET_STATUS register until the appropriate bit returns FLUSH_DONE.
3. Clear module bit in the CLK_RST_CONTROLLER_RST_DEVICES_* register to reset the module.
4. Set module bit in the CLK_RST_CONTROLLER_RST_DEVICES_* register to release the module reset.
5. Clear the appropriate FLUSH_ENABLE bit in the CLIENT_HOTRESET_CTRL register to allow transactions to flow.

Memory Client Hot Reset Control Register

Offset: 0x200 | Read/Write: R/W | Reset: 0bxxxxxxxxx0000x00x0x000000000000x

Bit	Reset	Description
17	DISABLE	VI_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
16	DISABLE	VDE_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
15	DISABLE	SATA_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
14	DISABLE	PPCS_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
13	DISABLE	NV2_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
12	DISABLE	NV_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
11	DISABLE	MPE_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
10	DISABLE	MPCORELP_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
9	DISABLE	MPCORE_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
8	DISABLE	ISP_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
7	DISABLE	HDA_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
6	DISABLE	HC_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
5	DISABLE	G2_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
4	DISABLE	EPP_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
3	DISABLE	DCB_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
2	DISABLE	DC_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
1	DISABLE	AVPC_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED
0	DISABLE	AFI_FLUSH_ENABLE: 0 = DISABLE 1 = ENABLE 0 = DISABLED 1 = ENABLED

20.11.1.37 MC_CLIENT_HOTRESET_STATUS_0

Contains one bit for each swname, indicating the status of any flush that has been requested in the CLIENT_HOTRESET_CTRL register.

NOTE: if no flush has been requested, register returns FLUSH_DONE.

Memory Client Hot Reset Status Register

Offset: 0x204 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
17	X	VI_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
16	X	VDE_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
15	X	SATA_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
14	X	PPCS_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
13	X	NV2_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
12	X	NV_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
11	X	MPE_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
10	X	MPCORELP_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
9	X	MPCORE_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
8	X	ISP_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
7	X	HDA_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
6	X	HC_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
5	X	G2_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
4	X	EPP_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
3	X	DCB_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
2	X	DC_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
1	X	AVPC_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS

Bit	Reset	Description
0	X	AFI_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS

20.11.1.38 MC_EMEM_ARB_ISOCHRONOUS_0_0

Controls whether a client is considered to be an isochronous client for EMEM arbitration. If a client is configured to be isochronous, the arbiter will favor expired isochronous requests over requests of any other type. This applies to arbitration for: best-bank arbiter (BA), activation arbiter (AA), direction arbiter (DA), and transaction arbiter (TA).

Per-Client Isochronous Settings

Offset: 0x208 | Read/Write: R/W | Reset: 0bx000xxx0000000110xxx000xx1111110

Bit	Reset	Description
31	DISABLE	ISO_SATAR: client satar is treated as an isochronous client 0 = DISABLE 1 = ENABLE
30	DISABLE	ISO_PPCSAHBSLVR: client ppcsahbslvr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
29	DISABLE	ISO_PPCSAHBDMAR: client ppcsahbdmar is treated as an isochronous client 0 = DISABLE 1 = ENABLE
28	DISABLE	ISO_MPECSRD: client mpecsrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE
27	DISABLE	ISO_MPEAMEMRD: client mpeamemrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE
26	DISABLE	ISO_MPE_IPRED: client mpe_ipred is treated as an isochronous client 0 = DISABLE 1 = ENABLE
25	DISABLE	ISO_IDXSRD2: client idxsrd2 is treated as an isochronous client 0 = DISABLE 1 = ENABLE
24	DISABLE	ISO_IDXSRD: client idxsrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE
23	DISABLE	ISO_HOST1XR: client host1xr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
22	DISABLE	ISO_HOST1XDMAR: client host1xdmar is treated as an isochronous client 0 = DISABLE 1 = ENABLE
21	DISABLE	ISO_HDAR: client hdar is treated as an isochronous client 0 = DISABLE 1 = ENABLE
20	DISABLE	ISO_G2DR: client g2dr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
19	DISABLE	ISO_FDCDRD2: client fdcdrd2 is treated as an isochronous client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	DISABLE	ISO_FDCDRD: client fdcdrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE
17	0x1	ISO_DISPLAYHCB: client displayhcb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
16	0x1	ISO_DISPLAYHC: client displayhc is treated as an isochronous client 0 = DISABLE 1 = ENABLE
15	DISABLE	ISO_AVPCARM7R: client avpcarm7r is treated as an isochronous client 0 = DISABLE 1 = ENABLE
14	DISABLE	ISO_AFIR: client afir is treated as an isochronous client 0 = DISABLE 1 = ENABLE
13	0x1	ISO_VIRUV: client viruv is treated as an isochronous client 0 = DISABLE 1 = ENABLE
12	DISABLE	ISO_MPEUNIFBR: client mpeunifbr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
11	DISABLE	ISO_G2SR: client g2sr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
10	DISABLE	ISO_G2PR: client g2pr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
9	DISABLE	ISO_EPPUP: client eppup is treated as an isochronous client 0 = DISABLE 1 = ENABLE
8	0x1	ISO_DISPLAY1BB: client display1bb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
7	0x1	ISO_DISPLAY1B: client display1b is treated as an isochronous client 0 = DISABLE 1 = ENABLE
6	0x1	ISO_DISPLAY0CB: client display0cb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
5	0x1	ISO_DISPLAY0C: client display0c is treated as an isochronous client 0 = DISABLE 1 = ENABLE
4	0x1	ISO_DISPLAY0BB: client display0bb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
3	0x1	ISO_DISPLAY0B: client display0b is treated as an isochronous client 0 = DISABLE 1 = ENABLE
2	0x1	ISO_DISPLAY0AB: client display0ab is treated as an isochronous client 0 = DISABLE 1 = ENABLE
1	0x1	ISO_DISPLAY0A: client display0a is treated as an isochronous client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x1	ISO_PTCR: client ptcr is treated as an isochronous client 0 = DISABLE 1 = ENABLE

20.11.1.39 MC_EMEM_ARB_ISOCHRONOUS_1_0

Controls whether a client is considered to be an isochronous client for EMEM arbitration. If a client is configured to be isochronous, the arbiter will favor expired isochronous requests over requests of any other type.

This applies to arbitration for: best-bank arbiter (BA), activation arbiter (AA), direction arbiter (DA), and transaction arbiter (TA).

Per-client isochronous settings

Offset: 0x20c | Read/Write: R/W | Reset: 0b00000000000000001111000000000000

Bit	Reset	Description
31	DISABLE	ISO_VDEDBGW: client vdedbgw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
30	DISABLE	ISO_VDEBSEVW: client vdebsevw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
29	DISABLE	ISO_SATAW: client sataw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
28	DISABLE	ISO_PPCSAHBSLVW: client ppcsahbslvw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
27	DISABLE	ISO_PPCSAHBDMAW: client ppcsahbdmaw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
26	DISABLE	ISO_MPECSWR: client mpecswr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
25	DISABLE	ISO_MPCOREW: client mpcorew is treated as an isochronous client 0 = DISABLE 1 = ENABLE
24	DISABLE	ISO_MPCORELPW: client mpcorelpw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
23	DISABLE	ISO_ISPW: client ispw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
22	DISABLE	ISO_HOST1XW: client host1xw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
21	DISABLE	ISO_HDAW: client hdaw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
20	DISABLE	ISO_FDCDWR2: client fdcdwr2 is treated as an isochronous client 0 = DISABLE 1 = ENABLE
19	DISABLE	ISO_FDCDWR: client fdcdwr is treated as an isochronous client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	DISABLE	ISO_AVPCARM7W: client avpcarm7w is treated as an isochronous client 0 = DISABLE 1 = ENABLE
17	DISABLE	ISO_AFIW: client afiw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
16	DISABLE	ISO_G2DW: client g2dw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
15	0x1	ISO_VIWY: client viwy is treated as an isochronous client 0 = DISABLE 1 = ENABLE
14	0x1	ISO_VIWV: client viwv is treated as an isochronous client 0 = DISABLE 1 = ENABLE
13	0x1	ISO_VIWU: client viwu is treated as an isochronous client 0 = DISABLE 1 = ENABLE
12	0x1	ISO_VIWSB: client viwsb is treated as an isochronous client 0 = DISABLE 1 = ENABLE
11	DISABLE	ISO_MPEUNIFBW: client mpeunifbw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
10	DISABLE	ISO_EPPY: client eppy is treated as an isochronous client 0 = DISABLE 1 = ENABLE
9	DISABLE	ISO_EPPV: client eppv is treated as an isochronous client 0 = DISABLE 1 = ENABLE
8	DISABLE	ISO_EPPU: client eppu is treated as an isochronous client 0 = DISABLE 1 = ENABLE
7	DISABLE	ISO_MPCORER: client mpcorer is treated as an isochronous client 0 = DISABLE 1 = ENABLE
6	DISABLE	ISO_MPCORELPR: client mpcorelpr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
5	DISABLE	ISO_VDETPER: client vdetper is treated as an isochronous client 0 = DISABLE 1 = ENABLE
4	DISABLE	ISO_VDEMCER: client vdemcer is treated as an isochronous client 0 = DISABLE 1 = ENABLE
3	DISABLE	ISO_VDEMBER: client vdember is treated as an isochronous client 0 = DISABLE 1 = ENABLE
2	DISABLE	ISO_VDEBSEVR: client vdebsevr is treated as an isochronous client 0 = DISABLE 1 = ENABLE
1	DISABLE	ISO_TEXSRD2: client texsrd2 is treated as an isochronous client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	DISABLE	ISO_TEXSRD: client texsrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE

20.11.1.40 MC_EMEM_ARB_ISOCHRONOUS_2_0

Controls whether a client is considered to be an isochronous client for EMEM arbitration. If a client is configured to be isochronous, the arbiter will favor expired isochronous requests over requests of any other type.

This applies to arbitration for: best-bank arbiter (BA), activation arbiter (AA), direction arbiter (DA), and transaction arbiter (TA).

Per-Client Isochronous Settings

Offset: 0x210 | Read/Write: R/W | Reset: 0bxxxxxxxxxx000000000000xxxxxxxx00

Bit	Reset	Description
1	DISABLE	ISO_VDETPMW: client vdetpmw is treated as an isochronous client 0 = DISABLE 1 = ENABLE
0	DISABLE	ISO_VDEMBEW: client vdembew is treated as an isochronous client 0 = DISABLE 1 = ENABLE

20.11.1.41 MC_EMEM_ARB_HYSTERESIS_0_0

Controls whether a client is considered to be a hysteresis client for EMEM arbitration. If a client is NOT marked hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Per-Client Hysteresis Settings

Offset: 0x218 | Read/Write: R/W | Reset: 0bx000xxx0000000110xxx000xx1111110

Bit	Reset	Description
31	DISABLE	HYST_SATAR: client satar is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
30	DISABLE	HYST_PPCSAHBSLVR: client ppcsahbslvr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
29	DISABLE	HYST_PPCSAHBDMAR: client ppcsahbdmar is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
28	DISABLE	HYST_MPECSRD: client mpecsrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
27	DISABLE	HYST_MPEAMEMRD: client mpeamemrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
26	DISABLE	HYST_MPE_IPRED: client mpe_ipred is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
25	DISABLE	HYST_IDXSRD2: client idxsrd2 is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
24	DISABLE	HYST_IDXSRD: client idxsrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
23	DISABLE	HYST_HOST1XR: client host1xr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
22	DISABLE	HYST_HOST1XDMAR: client host1xdmar is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
21	DISABLE	HYST_HDAR: client hdar is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
20	DISABLE	HYST_G2DR: client g2dr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
19	DISABLE	HYST_FDCDRD2: client fdcdrd2 is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
18	DISABLE	HYST_FDCDRD: client fdcdrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
17	0x1	HYST_DISPLAYHCB: client displayhcb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
16	0x1	HYST_DISPLAYHC: client displayhc is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
15	DISABLE	HYST_AVPCARM7R: client avpcarm7r is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
14	DISABLE	HYST_AFIR: client afir is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
13	0x1	HYST_VIRUV: client viruv is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
12	DISABLE	HYST_MPEUNIFBR: client mpeunifbr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
11	DISABLE	HYST_G2SR: client g2sr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
10	DISABLE	HYST_G2PR: client g2pr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
9	DISABLE	HYST_EPPUP: client eppup is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
8	0x1	HYST_DISPLAY1BB: client display1bb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
7	0x1	HYST_DISPLAY1B: client display1b is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x1	HYST_DISPLAY0CB: client display0cb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
5	0x1	HYST_DISPLAY0C: client display0c is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
4	0x1	HYST_DISPLAY0BB: client display0bb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
3	0x1	HYST_DISPLAY0B: client display0b is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
2	0x1	HYST_DISPLAY0AB: client display0ab is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
1	0x1	HYST_DISPLAY0A: client display0a is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
0	DISABLE	HYST_PTCR: client ptcr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

20.11.1.42 MC_EMEM_ARB_HYSTERESIS_1_0

Controls whether a client is considered to be an hysteresis client for EMEM arbitration. If a client is NOT marked hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Per-Client Hysteresis Settings

Offset: 0x21c | Read/Write: R/W | Reset: 0b00000000000000001111000000000000

Bit	Reset	Description
31	DISABLE	HYST_VDEDBGW: client vdedbgw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
30	DISABLE	HYST_VDEBSEVW: client vdebsevw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
29	DISABLE	HYST_SATAW: client sataw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
28	DISABLE	HYST_PPCSAHBSLVW: client ppcsahbslvw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
27	DISABLE	HYST_PPCSAHBDMAW: client ppcsahbdmaw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
26	DISABLE	HYST_MPECSWR: client mpecswr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
25	DISABLE	HYST_MPCOREW: client mpcorew is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
24	DISABLE	HYST_MPCORELPW: client mpcorelpw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
23	DISABLE	HYST_ISPW: client ispw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
22	DISABLE	HYST_HOST1XW: client host1xw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
21	DISABLE	HYST_HDAW: client hdaw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
20	DISABLE	HYST_FDCDWR2: client fdcdwr2 is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
19	DISABLE	HYST_FDCDWR: client fdcdwr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
18	DISABLE	HYST_AVPCARM7W: client avpcarm7w is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
17	DISABLE	HYST_AFIW: client afiw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
16	DISABLE	HYST_G2DW: client g2dw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
15	0x1	HYST_VIWY: client viwy is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
14	0x1	HYST_VIWV: client viwv is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
13	0x1	HYST_VIWU: client viwu is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
12	0x1	HYST_VIWSB: client viwsb is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
11	DISABLE	HYST_MPEUNIFBW: client mpeunifbw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
10	DISABLE	HYST_EPPY: client eppy is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
9	DISABLE	HYST_EPPV: client eppv is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
8	DISABLE	HYST_EPPU: client eppu is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
7	DISABLE	HYST_MPCORER: client mpcorer is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	DISABLE	HYST_MPCORELPR: client mpcorelpr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
5	DISABLE	HYST_VDETPER: client vdetper is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
4	DISABLE	HYST_VDEMCER: client vdemcer is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
3	DISABLE	HYST_VDEMBER: client vdember is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
2	DISABLE	HYST_VDEBSEVR: client vdebsevr is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
1	DISABLE	HYST_TEXSRD2: client texsrd2 is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
0	DISABLE	HYST_TEXSRD: client texsrd is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

20.11.1.43 MC_EMEM_ARB_HYSTERESIS_2_0

Controls whether a client is considered to be an hysteresis client for EMEM arbitration. If a client is NOT marked hysteresis, it will immediately cause the arbiter to turn off ts2aa_holdoff (see the comments for EMEM_ARB_OVERRIDE for further description of ts2aa_holdoff functionality).

Per-client hysteresis settings

Offset: 0x220 | Read/Write: R/W | Reset: 0bxxxxxxxxxx000000000000xxxxxxxx00

Bit	Reset	Description
1	DISABLE	HYST_VDETPMW: client vdetpmw is treated as a hysteresis client 0 = DISABLE 1 = ENABLE
0	DISABLE	HYST_VDEMBEW: client vdembew is treated as a hysteresis client 0 = DISABLE 1 = ENABLE

20.11.1.44 MC_SMMU_TRANSLATION_ENABLE_0_0

Used in addition to the global (MC_SMMU_CONFIG.SMMU_ENABLE) and per-module (MC_SMMU_*_ASID.*_SMMU_ENABLE) enables.

Access to this register is restricted to TrustZone-secured requestors.

Per-client SMMU translation enables

Offset: 0x228 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0bx111xxx1111111111xxx111xx111111x

Bit	Reset	Description
31	ENABLE	SMMU_SATAR_ENABLE: enable client satar to be translated by SMMU 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
30	ENABLE	SMMU_PPCSAHBSLVR_ENABLE: enable client ppcsahbslvr to be translated by SMMU 0 = DISABLE 1 = ENABLE
29	ENABLE	SMMU_PPCSAHBDMAR_ENABLE: enable client ppcsahbdmar to be translated by SMMU 0 = DISABLE 1 = ENABLE
28	ENABLE	SMMU_MPECSRD_ENABLE: enable client mpecsrd to be translated by SMMU 0 = DISABLE 1 = ENABLE
27	ENABLE	SMMU_MPEAMEMRD_ENABLE: enable client mpeamemrd to be translated by SMMU 0 = DISABLE 1 = ENABLE
26	ENABLE	SMMU_MPE_IPRED_ENABLE: enable client mpe_ipred to be translated by SMMU 0 = DISABLE 1 = ENABLE
25	ENABLE	SMMU_IDXSRD2_ENABLE: enable client idxsrd2 to be translated by SMMU 0 = DISABLE 1 = ENABLE
24	ENABLE	SMMU_IDXSRD_ENABLE: enable client idxsrd to be translated by SMMU 0 = DISABLE 1 = ENABLE
23	ENABLE	SMMU_HOST1XR_ENABLE: enable client host1xr to be translated by SMMU 0 = DISABLE 1 = ENABLE
22	ENABLE	SMMU_HOST1XDMAR_ENABLE: enable client host1xdmar to be translated by SMMU 0 = DISABLE 1 = ENABLE
21	ENABLE	SMMU_HDAR_ENABLE: enable client hdar to be translated by SMMU 0 = DISABLE 1 = ENABLE
20	ENABLE	SMMU_G2DR_ENABLE: enable client g2dr to be translated by SMMU 0 = DISABLE 1 = ENABLE
19	ENABLE	SMMU_FDCDRD2_ENABLE: enable client fdcdrd2 to be translated by SMMU 0 = DISABLE 1 = ENABLE
18	ENABLE	SMMU_FDCDRD_ENABLE: enable client fdcdrd to be translated by SMMU 0 = DISABLE 1 = ENABLE
17	ENABLE	SMMU_DISPLAYHCB_ENABLE: enable client displayhcb to be translated by SMMU 0 = DISABLE 1 = ENABLE
16	ENABLE	SMMU_DISPLAYHC_ENABLE: enable client displayhc to be translated by SMMU 0 = DISABLE 1 = ENABLE
15	ENABLE	SMMU_AVPCARM7R_ENABLE: enable client avpcarm7r to be translated by SMMU 0 = DISABLE 1 = ENABLE
14	ENABLE	SMMU_AFIR_ENABLE: enable client afir to be translated by SMMU 0 = DISABLE 1 = ENABLE
13	ENABLE	SMMU_VIRUV_ENABLE: enable client viruv to be translated by SMMU 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	ENABLE	SMMU_MPEUNIFBR_ENABLE: enable client mpeunifbr to be translated by SMMU 0 = DISABLE 1 = ENABLE
11	ENABLE	SMMU_G2SR_ENABLE: enable client g2sr to be translated by SMMU 0 = DISABLE 1 = ENABLE
10	ENABLE	SMMU_G2PR_ENABLE: enable client g2pr to be translated by SMMU 0 = DISABLE 1 = ENABLE
9	ENABLE	SMMU_EPPUP_ENABLE: enable client eppup to be translated by SMMU 0 = DISABLE 1 = ENABLE
8	ENABLE	SMMU_DISPLAY1BB_ENABLE: enable client display1bb to be translated by SMMU 0 = DISABLE 1 = ENABLE
7	ENABLE	SMMU_DISPLAY1B_ENABLE: enable client display1b to be translated by SMMU 0 = DISABLE 1 = ENABLE
6	ENABLE	SMMU_DISPLAY0CB_ENABLE: enable client display0cb to be translated by SMMU 0 = DISABLE 1 = ENABLE
5	ENABLE	SMMU_DISPLAY0C_ENABLE: enable client display0c to be translated by SMMU 0 = DISABLE 1 = ENABLE
4	ENABLE	SMMU_DISPLAY0BB_ENABLE: enable client display0bb to be translated by SMMU 0 = DISABLE 1 = ENABLE
3	ENABLE	SMMU_DISPLAY0B_ENABLE: enable client display0b to be translated by SMMU 0 = DISABLE 1 = ENABLE
2	ENABLE	SMMU_DISPLAY0AB_ENABLE: enable client display0ab to be translated by SMMU 0 = DISABLE 1 = ENABLE
1	ENABLE	SMMU_DISPLAY0A_ENABLE: enable client display0a to be translated by SMMU 0 = DISABLE 1 = ENABLE

20.11.1.45 MC_SMMU_TRANSLATION_ENABLE_1_0

Used in addition to the global (MC_SMMU_CONFIG.SMMU_ENABLE) and per-module (MC_SMMU_*_ASID.*_SMMU_ENABLE) enables.

Access to this register is restricted to TrustZone-secured requestors.

Per-client SMMU translation enables

Offset: 0x22c | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0b111111xx1111111111111111xx111111

Bit	Reset	Description
31	ENABLE	SMMU_VDEDBGW_ENABLE: enable client vdedbgw to be translated by SMMU 0 = DISABLE 1 = ENABLE
30	ENABLE	SMMU_VDEBSEVW_ENABLE: enable client vdebsevw to be translated by SMMU 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
29	ENABLE	SMMU_SATAW_ENABLE: enable client sataw to be translated by SMMU 0 = DISABLE 1 = ENABLE
28	ENABLE	SMMU_PPCSAHBSLVW_ENABLE: enable client ppcsahbslvw to be translated by SMMU 0 = DISABLE 1 = ENABLE
27	ENABLE	SMMU_PPCSAHBDMAW_ENABLE: enable client ppcsahbdmaw to be translated by SMMU 0 = DISABLE 1 = ENABLE
26	ENABLE	SMMU_MPECSWR_ENABLE: enable client mpecswr to be translated by SMMU 0 = DISABLE 1 = ENABLE
23	ENABLE	SMMU_ISPW_ENABLE: enable client ispw to be translated by SMMU 0 = DISABLE 1 = ENABLE
22	ENABLE	SMMU_HOST1XW_ENABLE: enable client host1xw to be translated by SMMU 0 = DISABLE 1 = ENABLE
21	ENABLE	SMMU_HDAW_ENABLE: enable client hdaw to be translated by SMMU 0 = DISABLE 1 = ENABLE
20	ENABLE	SMMU_FDCDWR2_ENABLE: enable client fdcdwr2 to be translated by SMMU 0 = DISABLE 1 = ENABLE
19	ENABLE	SMMU_FDCDWR_ENABLE: enable client fdcdwr to be translated by SMMU 0 = DISABLE 1 = ENABLE
18	ENABLE	SMMU_AVPCARM7W_ENABLE: enable client avpcarm7w to be translated by SMMU 0 = DISABLE 1 = ENABLE
17	ENABLE	SMMU_AFIW_ENABLE: enable client afiw to be translated by SMMU 0 = DISABLE 1 = ENABLE
16	ENABLE	SMMU_G2DW_ENABLE: enable client g2dw to be translated by SMMU 0 = DISABLE 1 = ENABLE
15	ENABLE	SMMU_VIWY_ENABLE: enable client viwy to be translated by SMMU 0 = DISABLE 1 = ENABLE
14	ENABLE	SMMU_VIWV_ENABLE: enable client viwv to be translated by SMMU 0 = DISABLE 1 = ENABLE
13	ENABLE	SMMU_VIWU_ENABLE: enable client viwu to be translated by SMMU 0 = DISABLE 1 = ENABLE
12	ENABLE	SMMU_VIWSB_ENABLE: enable client viwsb to be translated by SMMU 0 = DISABLE 1 = ENABLE
11	ENABLE	SMMU_MPEUNIFBW_ENABLE: enable client mpeunifbw to be translated by SMMU 0 = DISABLE 1 = ENABLE
10	ENABLE	SMMU_EPPY_ENABLE: enable client eppy to be translated by SMMU 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	ENABLE	SMMU_EPPV_ENABLE: enable client eppv to be translated by SMMU 0 = DISABLE 1 = ENABLE
8	ENABLE	SMMU_EPPU_ENABLE: enable client eppu to be translated by SMMU 0 = DISABLE 1 = ENABLE
5	ENABLE	SMMU_VDETPER_ENABLE: enable client vdetper to be translated by SMMU 0 = DISABLE 1 = ENABLE
4	ENABLE	SMMU_VDEMCER_ENABLE: enable client vdemcer to be translated by SMMU 0 = DISABLE 1 = ENABLE
3	ENABLE	SMMU_VDEMBER_ENABLE: enable client vdember to be translated by SMMU 0 = DISABLE 1 = ENABLE
2	ENABLE	SMMU_VDEBSEVR_ENABLE: enable client vdebsevr to be translated by SMMU 0 = DISABLE 1 = ENABLE
1	ENABLE	SMMU_TEXSRD2_ENABLE: enable client texsrd2 to be translated by SMMU 0 = DISABLE 1 = ENABLE
0	ENABLE	SMMU_TEXSRD_ENABLE: enable client texsrd to be translated by SMMU 0 = DISABLE 1 = ENABLE

20.11.1.46 MC_SMMU_TRANSLATION_ENABLE_2_0

Used in addition to the global (MC_SMMU_CONFIG.SMMU_ENABLE) and per-module (MC_SMMU_*_ASID.*_SMMU_ENABLE) enables.

Access to this register is restricted to TrustZone-secured requestors.

Per-client SMMU translation enables

Offset: 0x230 | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0bxxxxxxxxxx11xx11111111xxxxxxxx11

Bit	Reset	Description
1	ENABLE	SMMU_VDETPMW_ENABLE: enable client vdetpmw to be translated by SMMU 0 = DISABLE 1 = ENABLE
0	ENABLE	SMMU_VDEMBEW_ENABLE: enable client vdembew to be translated by SMMU 0 = DISABLE 1 = ENABLE

20.11.1.47 MC_SMMU_AFI_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU AFI ASID Assignment Register

Offset: 0x238 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	AFI_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	AFI_ASID: ASID to use for translation, if enabled

20.11.1.48 MC_SMMU_AVPC_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU AVPC ASID Assignment Register

Offset: 0x23c | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	AVPC_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	AVPC_ASID: ASID to use for translation, if enabled

20.11.1.49 MC_SMMU_DC_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU DC ASID Assignment Register

Offset: 0x240 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	DC_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	DC_ASID: ASID to use for translation, if enabled

20.11.1.50 MC_SMMU_DCB_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU DCB ASID Assignment Register

Offset: 0x244 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	DCB_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	DCB_ASID: ASID to use for translation, if enabled

20.11.1.51 MC_SMMU_EPP_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU EPP ASID Assignment Register

Offset: 0x248 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	EPP_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	EPP_ASID: ASID to use for translation, if enabled

20.11.1.52 MC_SMMU_G2_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU G2 ASID Assignment Register

Offset: 0x24c | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	G2_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	G2_ASID: ASID to use for translation, if enabled

20.11.1.53 MC_SMMU_HC_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU HC ASID Assignment Register

Offset: 0x250 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	HC_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	HC_ASID: ASID to use for translation, if enabled

20.11.1.54 MC_SMMU_HDA_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU HDA ASID Assignment Register

Offset: 0x254 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	HDA_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	HDA_ASID: ASID to use for translation, if enabled

20.11.1.55 MC_SMMU_ISP_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU ISP ASID Assignment Register

Offset: 0x258 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	ISP_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	ISP_ASID: ASID to use for translation, if enabled

20.11.1.56 MC_SMMU_MPE_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU MPE ASID Assignment Register

Offset: 0x264 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	MPE_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	MPE_ASID: ASID to use for translation, if enabled

20.11.1.57 MC_SMMU_NV_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU NV ASID Assignment Register

Offset: 0x268 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	NV_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	NV_ASID: ASID to use for translation, if enabled

20.11.1.58 MC_SMMU_NV2_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU NV2 ASID Assignment Register

Offset: 0x26c | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	NV2_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	NV2_ASID: ASID to use for translation, if enabled

20.11.1.59 MC_SMMU_PPCS_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU PPCS ASID Assignment Register

Offset: 0x270 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	PPCS_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	PPCS_ASID: ASID to use for translation, if enabled

20.11.1.60 MC_SMMU_SATA_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU SATA ASID Assignment Register

Offset: 0x278 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	SATA_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	SATA_ASID: ASID to use for translation, if enabled

20.11.1.61 MC_SMMU_VDE_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU VDE ASID Assignment Register

Offset: 0x27c | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	VDE_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	VDE_ASID: ASID to use for translation, if enabled

20.11.1.62 MC_SMMU_VI_ASID_0

Enables or disables SMMU translation for the module and if enabled sets the ASID for translation.

This register is not TrustZone secure, but non-secure writes to this register will be dropped if either the current or new value of the ASID field is for a secure ASID (regardless of the current or new value of the SMMU_ENABLE field.

SMMU VI ASID Assignment Register

Offset: 0x280 | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	VI_SMMU_ENABLE: if set, translation is enabled for this module 0 = DISABLE 1 = ENABLE
1:0	0x0	VI_ASID: ASID to use for translation, if enabled

20.11.1.63 MC_A9LP_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests, the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client a9lp

Offset: 0x29c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_A9LP

20.11.1.64 MC_AHB_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests; the client still has 1 request.

Additional snap arbiter levels for partition client AHB

Offset: 0x2a0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_AHB

20.11.1.65 MC_APB_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests, the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client APB

Offset: 0x2a4 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_APB

20.11.1.66 MC_AVP_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests, the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client AVP

Offset: 0x2a8 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_AVP

20.11.1.67 MC_DIS_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests; the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client DIS

Offset: 0x2ac | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11

Bit	Reset	Description
1:0	0x3	SNAP_LEVELS_DIS

20.11.1.68 MC_HEG_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests; the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client HEG

Offset: 0x2b0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_HEG

20.11.1.69 MC_ME_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests, the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client ME

Offset: 0x2b4 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_ME

20.11.1.70 MC_PCX_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests, the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client PCX

Offset: 0x2b8 | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_PCX

20.11.1.71 MC_PI_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests, the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client PI

Offset: 0x2bc | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_PI

20.11.1.72 MC_SAX_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests, the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client SAX

Offset: 0x2c0 | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_SAX

20.11.1.73 MC_TDA_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests; the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client TDA

Offset: 0x2c4 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_TDA

20.11.1.74 MC_TDA2_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests; the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client TDA2

Offset: 0x2c8 | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_TDA2

20.11.1.75 MC_TDB_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests; the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client TDB

Offset: 0x2cc | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_TDB

20.11.1.76 MC_TDB2_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests; the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client TDB2

Offset: 0x2d0 | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_TDB2

20.11.1.77 MC_VD_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests; the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client VD

Offset: 0x2d4 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_VD

20.11.1.78 MC_VE_EXTRA_SNAP_LEVELS_0

Each additional level provides an extra request to the snap arbiter. For n requesters, each will get 1/n of the bandwidth. Adding a second level for one client will give that client 2/n+1 of the bandwidth.

Note that a value of 0 means no 'additional' requests; the client still has 1 request.

Additional Snap Arbiter Levels for Partition Client VE

Offset: 0x2d8 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_VE

20.11.1.79 MC_LATENCY_ALLOWANCE_AFI_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for AFI clients

Offset: 0x2e0 | Read/Write: R/W | Reset: 0b00001100xxxxxxxx00010000

Bit	Reset	Description
23:16	0xc	ALLOWANCE_AFIW
7:0	0x10	ALLOWANCE_AFIR

20.11.1.80 MC_LATENCY_ALLOWANCE_AVPC_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for AVPC clients

Offset: 0x2e4 | Read/Write: R/W | Reset: 0bxxxxxxxx00001110xxxxxxxx00000100

Bit	Reset	Description
23:16	0xe	ALLOWANCE_AVPCARM7W
7:0	0x4	ALLOWANCE_AVPCARM7R

20.11.1.81 MC_LATENCY_ALLOWANCE_DC_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for DC clients

Offset: 0x2e8 | Read/Write: R/W | Reset: 0bxxxxxxxx01001110xxxxxxxx01001110

Bit	Reset	Description
23:16	0x4e	ALLOWANCE_DISPLAY0B
7:0	0x4e	ALLOWANCE_DISPLAY0A

20.11.1.82 MC_LATENCY_ALLOWANCE_DC_1_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for DC clients

Offset: 0x2ec | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx01001110

Bit	Reset	Description
23:16	0x4e	ALLOWANCE_DISPLAY1B
7:0	0x4e	ALLOWANCE_DISPLAY0C

20.11.1.83 MC_LATENCY_ALLOWANCE_DC_2_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for DC clients

Offset: 0x2f0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx01101000

Bit	Reset	Description
7:0	0xff	ALLOWANCE_DISPLAYHC

20.11.1.84 MC_LATENCY_ALLOWANCE_DCB_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for DCB clients

Offset: 0x2f4 | Read/Write: R/W | Reset: 0bxxxxxxxx01001110xxxxxxxx01001110

Bit	Reset	Description
23:16	0x4e	ALLOWANCE_DISPLAY0BB
7:0	0x4e	ALLOWANCE_DISPLAY0AB

20.11.1.85 MC_LATENCY_ALLOWANCE_DCB_1_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for DCB clients

Offset: 0x2f8 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx01001110

Bit	Reset	Description
23:16	0x4e	ALLOWANCE_DISPLAY1BB
7:0	0x4e	ALLOWANCE_DISPLAY0CB

20.11.1.86 MC_LATENCY_ALLOWANCE_DCB_2_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for DCB clients

Offset: 0x2fc | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx01101000

Bit	Reset	Description
7:0	0xff	ALLOWANCE_DISPLAYHCB

20.11.1.87 MC_LATENCY_ALLOWANCE_EPP_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for EPP clients

Offset: 0x300 | Read/Write: R/W | Reset: 0bxxxxxxxx01101100xxxxxxxx00110011

Bit	Reset	Description
23:16	0x6c	ALLOWANCE_EPPU
7:0	0x17	ALLOWANCE_EPPUP

20.11.1.88 MC_LATENCY_ALLOWANCE_EPP_1_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for EPP clients

Offset: 0x304 | Read/Write: R/W | Reset: 0bxxxxxxxx01101100xxxxxxxx01101100

Bit	Reset	Description
23:16	0x6c	ALLOWANCE_EPPY
7:0	0x6c	ALLOWANCE_EPPV

20.11.1.89 MC_LATENCY_ALLOWANCE_G2_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for G2 clients

Offset: 0x308 | Read/Write: R/W | Reset: 0bxxxxxxxx00001001xxxxxxxx00001001

Bit	Reset	Description
23:16	0x9	ALLOWANCE_G2SR
7:0	0x9	ALLOWANCE_G2PR

20.11.1.90 MC_LATENCY_ALLOWANCE_G2_1_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for G2 clients

Offset: 0x30c | Read/Write: R/W | Reset: 0bxxxxxxxx00001001xxxxxxxx00001010

Bit	Reset	Description
23:16	0x9	ALLOWANCE_G2DW
7:0	0xa	ALLOWANCE_G2DR

20.11.1.91 MC_LATENCY_ALLOWANCE_HC_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for HC clients

Offset: 0x310 | Read/Write: R/W | Reset: 0bxxxxxxxx10100101xxxxxxxx00010000

Bit	Reset	Description
23:16	0x50	ALLOWANCE_HOST1XR
7:0	0x5	ALLOWANCE_HOST1XDMAR

20.11.1.92 MC_LATENCY_ALLOWANCE_HC_1_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for HC clients

Offset: 0x314 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx00100101

Bit	Reset	Description
7:0	0x10	ALLOWANCE_HOST1XW

20.11.1.93 MC_LATENCY_ALLOWANCE_HDA_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for HDA clients

Offset: 0x318 | Read/Write: R/W | Reset: 0bxxxxxxxx11111111xxxxxxxx11111111

Bit	Reset	Description
23:16	0xff	ALLOWANCE_HDAW
7:0	0xff	ALLOWANCE_HDAR

20.11.1.94 MC_LATENCY_ALLOWANCE_ISP_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for ISP clients

Offset: 0x31c Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx11111111

Bit	Reset	Description
7:0	0xff	ALLOWANCE_ISPW

20.11.1.95 MC_LATENCY_ALLOWANCE_MPCORE_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for MPCORE clients

Offset: 0x320 | Read/Write: R/W | Reset: 0bxxxxxxxx00001110xxxxxxxx00000100

Bit	Reset	Description
23:16	0xe	ALLOWANCE_MPCOREW
7:0	0x4	ALLOWANCE_MPCORER

20.11.1.96 MC_LATENCY_ALLOWANCE_MPCORELP_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for MPCORELP clients

Offset: 0x324 | Read/Write: R/W | Reset: 0bxxxxxxxx00001110xxxxxxxx00000100

Bit	Reset	Description
23:16	0xe	ALLOWANCE_MPCORELPW
7:0	0x4	ALLOWANCE_MPCORELPR

20.11.1.97 MC_LATENCY_ALLOWANCE_MPE_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for MPE clients

Offset: 0x328 | Read/Write: R/W | Reset: 0bxxxxxxxx10000000xxxxxxxx10000000

Bit	Reset	Description
23:16	0x80	ALLOWANCE_MPE_IPRED
7:0	0x50	ALLOWANCE_MPEUNIFBR

20.11.1.98 MC_LATENCY_ALLOWANCE_MPE_1_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for MPE clients

Offset: 0x32c | Read/Write: R/W | Reset: 0b11111111xxxxxxxx01000010

Bit	Reset	Description
23:16	0xff	ALLOWANCE_MPECSRD
7:0	0x42	ALLOWANCE_MPEAMEMRD

20.11.1.99 MC_LATENCY_ALLOWANCE_MPE_2_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for MPE clients

Offset: 0x330 | Read/Write: R/W | Reset: 0b11111111xxxxxxxx00010011

Bit	Reset	Description
23:16	0xff	ALLOWANCE_MPECSWR
7:0	0x13	ALLOWANCE_MPEUNIFBW

20.11.1.100 MC_LATENCY_ALLOWANCE_NV_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for NV clients

Offset: 0x334 | Read/Write: R/W | Reset: 0bxxxxxxxx00001011xxxxxxxx00001100

Bit	Reset	Description
23:16	0x13	ALLOWANCE_IDXSRD
7:0	0xa	ALLOWANCE_FDCDRD

20.11.1.101 MC_LATENCY_ALLOWANCE_NV_1_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for NV clients

Offset: 0x338 | Read/Write: R/W | Reset: 0bxxxxxxxx00010000xxxxxxxx00001100

Bit	Reset	Description
23:16	0xa	ALLOWANCE_FDCDWR
7:0	0x13	ALLOWANCE_TEXSRD

20.11.1.102 MC_LATENCY_ALLOWANCE_NV2_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for NV2 clients

Offset: 0x33c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx00001100

Bit	Reset	Description
23:16	0x13	ALLOWANCE_IDXSRD2
7:0	0xa	ALLOWANCE_FDCDRD2

20.11.1.103 MC_LATENCY_ALLOWANCE_NV2_1_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for NV2 clients

Offset: 0x340 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx00010000

Bit	Reset	Description
23:16	0xa	ALLOWANCE_FDCDWR2
7:0	0x13	ALLOWANCE_TEXSRD2

20.11.1.104 MC_LATENCY_ALLOWANCE_PPCS_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for PPCS clients

Offset: 0x344 | Read/Write: R/W | Reset: 0bxxxxxxxx11101000xxxxxxxx01010000

Bit	Reset	Description
23:16	0x12	ALLOWANCE_PPCSAHBSLVR
7:0	0x10	ALLOWANCE_PPCSAHBDMAR

20.11.1.105 MC_LATENCY_ALLOWANCE_PPCS_1_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for PPCS clients

Offset: 0x348 | Read/Write: R/W | Reset: 0bxxxxxxxx11101000xxxxxxxx10100101

Bit	Reset	Description
23:16	0x6	ALLOWANCE_PPCSAHBSLVW
7:0	0x10	ALLOWANCE_PPCSAHBDMAW

20.11.1.106 MC_LATENCY_ALLOWANCE_PTC_0_0

NV_PTCR2MC_SR_TRAFFIC_TYPE not found

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for PTC clients

Offset: 0x34c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
7:0	0x0	ALLOWANCE_PTCR

20.11.1.107 MC_LATENCY_ALLOWANCE_SATA_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for SATA clients

Offset: 0x350 | Read/Write: R/W | Reset: 0b00110011xxxxxxxx00110011

Bit	Reset	Description
23:16	0x33	ALLOWANCE_SATAW
7:0	0x33	ALLOWANCE_SATAR

20.11.1.108 MC_LATENCY_ALLOWANCE_VDE_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for VDE clients

Offset: 0x354 | Read/Write: R/W | Reset: 0bxxxxxxxx11111111xxxxxxxx11111111

Bit	Reset	Description
23:16	0xd0	ALLOWANCE_VDEMBER
7:0	0xff	ALLOWANCE_VDEBSEVR

20.11.1.109 MC_LATENCY_ALLOWANCE_VDE_1_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for VDE clients

Offset: 0x358 | Read/Write: R/W | Reset: 0bxxxxxxxx11101110xxxxxxxx10111000

Bit	Reset	Description
23:16	0x74	ALLOWANCE_VDETPER
7:0	0x2a	ALLOWANCE_VDEMCER

20.11.1.110 MC_LATENCY_ALLOWANCE_VDE_2_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for VDE clients

Offset: 0x35c | Read/Write: R/W | Reset: 0bxxxxxxxx11111111xxxxxxxx11111111

Bit	Reset	Description
23:16	0xff	ALLOWANCE_VDEDBGW
7:0	0xff	ALLOWANCE_VDEBSEVW

20.11.1.111 MC_LATENCY_ALLOWANCE_VDE_3_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for VDE clients

Offset: 0x360 | Read/Write: R/W | Reset: 0bxxxxxxxx01011001xxxxxxxx10001001

Bit	Reset	Description
23:16	0x2a	ALLOWANCE_VDETPMW
7:0	0x42	ALLOWANCE_VDEMBEW

20.11.1.112 MC_LATENCY_ALLOWANCE_VI_0_0

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Latency allowance settings for VI clients

Offset: 0x364 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx01000111

Bit	Reset	Description
23:16	0x12	ALLOWANCE_VIWSB
7:0	0x2c	ALLOWANCE_VIRUV

20.11.1.113 MC_LATENCY_ALLOWANCE_VI_1_0

Latency allowance settings for VI clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Offset: 0x368 | Read/Write: R/W | Reset: 0bxxxxxxxx11111111xxxxxxxx11111111

Bit	Reset	Description
23:16	0xb2	ALLOWANCE_VIWV
7:0	0xb2	ALLOWANCE_VIWU

20.11.1.114 MC_LATENCY_ALLOWANCE_VI_2_0

Latency allowance settings for VI clients

The number of ticks a request from a particular client may wait in the EMEM arbiter before it becomes a high-priority request.

See EMEM_ARB_CFG for definition of the units 'ticks'.

This value should be set with consideration for the module's latency tolerance and internal buffering capabilities.

Offset: 0x36c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx01000111

Bit	Reset	Description
7:0	0x12	ALLOWANCE_VIWY

20.11.1.115 MC_PTSA_GRANT_DECREMENT_0

The amount that the PTSA DDAs are decremented each cycle there is a grant should be programmed to the fraction of total memory bandwidth / maximum total memory bandwidth.

Offset: 0x960 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx100000000

Bit	Reset	Description
8	0x1	PTSA_GRANT_DECREMENT_INT: Integer portion (0 or 1)
7:0	0x0	PTSA_GRANT_DECREMENT_FRAC: Fractional portion

20.11.2 EMC Registers

> **Note:** Please use EMC_WINCR definition to specify incr1/incr4 based on the project defines.

USAGE NOTE: Many EMC register fields are shadowed:

- Writes to shadowed register fields update the shadow copy (this is default, assumes DBG.WRITE_MUX==ASSEMBLY).
- Reads to shadowed register fields return the currently-active copy (this is default, assumes DBG.READ_MUX==ACTIVE).

This allows a new set of frequency-dependent timing parameters to be written to the shadow registers while memory traffic is ongoing, then when the parameters are completely written, the EMC hardware can perform a timing-safe switch.

Such switches can be triggered via two methods: TIMING_CONTROL.TIMING_UPDATE (generally used during initialization), or the automatic CAR/EMC handshake on a clock-frequency or divider change (assuming CFG_2.CLKCHANGE_REQ_ENABLE==ENABLED).

Registers that are shadowed are marked by a comment:

```
// This register is shadowed: see usage note at top.
```

Occasionally, only certain fields in the register will be shadowed, if so they are noted after the above comment.

USAGE NOTE: Many EMC registers play crucial roles in warm boot (also known as "wake from LP0") and cold boot (also known as "power up") sequences. Suggested actions for the Boot ROM/Bootloader are noted in comments labelled "Boot requirements: ...".

USAGE NOTE: The EMC register fields to be stored in the PMC must have "[PMC]", "[PMC2]", or "[PMC3]".in their comments. ([PMC2] registers are packed/unpacked after [PMC] register group in software code. [PMC3] is even later.) The fields that need to be stored in PMC Secure Scratch registers must have [PMC_SECURE] in their comments.

20.11.2.1 EMC_INTSTATUS_0

Interrupt Status Register

Clear on 1-write. Init value is clear.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000x0x)

Bit	Reset	Description
9	CLEAR	DLL_LOCK_TIMEOUT_INT: Indicates a DLL lock timeout has occurred. 0 = CLEAR 1 = SET
8	CLEAR	CCFIFO_OVERFLOW_INT: Clock change FIFO overflow. 0 = CLEAR 1 = SET
7	CLEAR	DLL_ALARM_INT: Indicates a DLL alarm has been set 0 = CLEAR 1 = SET
6	CLEAR	ACCESS_TO_SR_DPD_DEV_INT: Indicates a system attempt to access a self-refresh/deep-powered-down device. 0 = CLEAR 1 = SET
5	CLEAR	MRR_DIVLD_INT: LPDDR2 MRR data is available to be read. 0 = CLEAR 1 = SET
4	CLEAR	CLKCHANGE_COMPLETE_INT: CAR/EMC clock-change handshake complete. 0 = CLEAR 1 = SET
3	CLEAR	REFRESH_OVERFLOW_INT: Refresh request overflow timeout. 0 = CLEAR 1 = SET
1	CLEAR	EXT_INTR_IN_INT: External interrupt input (from the next EMC) 0 = CLEAR 1 = SET

20.11.2.2 EMC_INTMASK_0

Interrupt Mask Register

Init value is masked.

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000x0x)

Bit	Reset	Description
9	MASKED	DLL_LOCK_TIMEOUT_INTMASK: Mask for DLL lock timeout interrupt. 0 = MASKED 1 = UNMASKED
8	MASKED	CCFIFO_OVERFLOW_INTMASK: Mask for clock change FIFO overflow. 0 = MASKED 1 = UNMASKED
7	MASKED	DLL_ALARM_INTMASK: Mask for DLL alarm 0 = MASKED 1 = UNMASKED
6	MASKED	ACCESS_TO_SR_DPD_DEV_INTMASK: Mask for access a self-refresh/deep-powered-down device interrupt. 0 = MASKED 1 = UNMASKED
5	MASKED	MRR_DIVLD_INTMASK: Mask for MRR data available. 0 = MASKED 1 = UNMASKED
4	MASKED	CLKCHANGE_COMPLETE_INTMASK: Mask for CAR/EMC clock-change handshake complete. 0 = MASKED 1 = UNMASKED
3	MASKED	REFRESH_OVERFLOW_INTMASK: Mask for refresh request overflow timeout. 0 = MASKED 1 = UNMASKED
1	MASKED	EXT_INTR_IN_INTMASK: Mask for external interrupt input 0 = MASKED 1 = UNMASKED

20.11.2.3 EMC_DBG_0

Debug Register

The DBG register is used to reconfigure the EMC during debug or chip testing.

Boot Requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x8 | Read/Write: R/W | Reset: 0x01000400 (0bxxxxxxx1xxxxxxxxxx00x10xxxxx0000)

Bit	Reset	Description
24	ENABLED	CFG_PRIORITY: determines the priority of cfg accesses to the DRAM. Setting this register to ENABLED gives DRAM config cycles (refresh, mrs, emrs, etc.) higher priority over real time requestors. The DISABLED setting gives the real time requestors higher priority than DRAM config cycles. Do not program to DISABLED unless for debugging. 0 = DISABLED 1 = ENABLED
13	DISABLED	SUPPRESS_WRITE_CMD: suppress write command sent to DRAM

Bit	Reset	Description
		0 = DISABLED 1 = ENABLED
12	DISABLED	SUPPRESS_READ_CMD: suppress read command sent to DRAM 0 = DISABLED 1 = ENABLED
10	ENABLED	AP_REQ_BUSY_CTRL: determines whether the busy signal from the auto-precharge cancellation (APC) FIFO is allowed to stall requests to the EMC. This field should usually be set to ENABLED 0 = DISABLED 1 = ENABLED
9	MANAGED	READ_DQM_CTRL: If set to MANAGED, EMC only turns them on when necessary. If set to ALWAYS_ON, the DQM signals are enabled during non-write operation. This field should usually be set to MANAGED 0 = MANAGED 1 = ALWAYS_ON
2	DISABLED	FORCE_UPDATE: causes the active state to get updated with the assembly state immediately upon setting this register. 0 = DISABLED 1 = ENABLED
1	ASSEMBLY	WRITE_MUX: controls whether writes to the configuration registers are done from the assembly or active state. 0 = ASSEMBLY 1 = ACTIVE
0	ACTIVE	READ_MUX: controls whether reads to the configuration registers are done from the assembly or active state. 0 = ACTIVE 1 = ASSEMBLY

20.11.2.4 EMC_CFG_0

Configuration Register

The CFG register is used to configure the external memory interface.

Note: For each EMC2PMACRO_CFG_BYPASS_DATAPIPE*=0, the timing parameters (WDV, RDV, CTT, PUTERM, PUTERM_EXTRA,QUSE, QUSE_EXTRA, QRST, EINPUT, IBDLY) must be subtracted by 1 to compensate. In other words, if both *_BYPASS_DATAPIPE* are set to 0, the above timing parameters need to be subtracted by 2.

For EMC2PMACRO_CFG_BYPASS_ADDRPIPE=0, the above timing parameters need to be added by 1.

Boot Requirements

- This register should be parameterized in the BCT and written by the BootROM during cold boot.
- This register should be saved in the scratch registers and restored by the BootROM during warm boot.

Offset: 0xc | Read/Write: R/W | Reset: 0x03c0000e (0b0000xx11110000xxxxxxxxxx000111x)

Bit	Reset	Description
31	DISABLED	DRAM_CLKSTOP_PD: [PMC] clock stop is only allowed to happen if the DRAM is in active/precharge powerdown (for all CKE bits associated with clock) 0 = DISABLED 1 = ENABLED
30	DISABLED	DRAM_CLKSTOP_SR: [PMC] clockstop is only allowed to happen if the DRAM is in self-refresh (for all CKE bits associated with clock) 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
29	NO_POWERDOWN	DRAM_ACPD: [PMC] allows the DRAM controller to perform opportunistic active powerdown control using the CKE pin on the DRAM. The behavior of the powerdown control logic is controlled by the PDEX2* and *2PDEN registers. The value of DRAM_ACPD should only be changed when CKE is low, e.g., during software-controlled self-refresh or before DRAM initialization. 0 = NO_POWERDOWN 1 = ACTIVE_POWERDOWN
28	DISABLED	DYN_SELF_REF: [PMC] 0 = DISABLED 1 = ENABLED
25	ENABLED	AUTO_PRE_WR: [PMC] enable using MC auto-precharge indication for writes. This bits, when set to DISABLE, will override the settings in the MC register (EMC ignore the auto-precharge indication from MC). Otherwise, they permit clients to make either auto or explicit precharge requests depend on the state of MAN_PRE_WR. 0 = DISABLED 1 = ENABLED
24	ENABLED	AUTO_PRE_RD: [PMC] enable using MC auto-precharge indication for reads. This bits, when set to DISABLE, will override the settings in the MC register (EMC ignores the auto-precharge indication from MC). Otherwise, they permit clients to make either auto or explicit precharge requests depend on the state of MAN_PRE_RD. 0 = DISABLED 1 = ENABLED
23	ENABLED	MAN_PRE_WR: [PMC] enable explicit-precharge in the EMC for writes. When this bit is enabled (and AUTO_PRE_WR=enable), EMC will issue an explicit precharge command after the memory write command. If this bit is disabled (and AUTO_PRE_WR=enable), EMC will issue an auto-precharge with the memory write command. 0 = DISABLED 1 = ENABLED
22	ENABLED	MAN_PRE_RD: [PMC] enable explicit-precharge in the EMC for reads. When this bit is enabled (and AUTO_PRE_RD=enable), EMC will issue an explicit precharge command after the memory read command. If this bit is disabled (and AUTO_PRE_RD=enable), EMC will issue an auto-precharge with the memory read command. 0 = DISABLED 1 = ENABLED
21	DISABLED	PERIODIC_QRST: [PMC] specifies whether or not to periodic reset the FBIO read-data FIFO during normal operation. The periodic resets can be used for graceful recovery from an intermittent failure condition; only the initial reset is absolutely required. Note: For LPDDRx and MRR (if ever used), this bit should be set to enable. 0 = DISABLED 1 = ENABLED
20	DISABLED	EN_DYNAMIC_PUTERM: [PMC] enable dynamic puterm. 0 = DISABLED 1 = ENABLED
19	DISABLED	DLY_WR_DQ_HALF_CLOCK: [PMC] 1 = delay write data by 1/2 clock. 0 = DISABLED 1 = ENABLED
18	DISABLED	DSR_VTTGEN_DRV_EN: [PMC] enable VTTGEN controls (via DSR_VTTGEN_DRV/TXDSRVTTGEN registers) during DSR. 0 = DISABLED 1 = ENABLED
6	0x0	INVERT_DQM: [PMC] 1 = invert DQM polarity.
5	0x0	WAIT_FOR_DISPLAYB_READY_B4_CC: [PMC] 1 = wait for displayb ready event to be asserted before acknowledging clock change.
4	0x0	WAIT_FOR_DISPLAY_READY_B4_CC: [PMC] 1 = wait for display ready event to be asserted before acknowledging clock change.

Bit	Reset	Description
3	0x1	EMC2PMACRO_CFG_BYPASS_DATAPIPE2: [PMC] 1 = bypass data pipeline stage2 (closer to I/O) between EMC and data pad-macro.
2	0x1	EMC2PMACRO_CFG_BYPASS_DATAPIPE1: [PMC] 1 = bypass data pipeline stage1 (closer to core) between EMC and data pad-macro.
1	0x1	EMC2PMACRO_CFG_BYPASS_ADDRPIPE: [PMC] 1 = bypass address pipeline stage between EMC and address pad-macro.

20.11.2.5 EMC_ADR_CFG_0

External Memory Address Configuration, System

The ADR_CFG register is used to specify the number of DRAM devices. DRAM density and geometry parameters are specified by the EMEM_ADR_CFG* registers in MC.

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	Reset	Description
7	DISABLED	EMEM_ROW_MSB_ON_CS1: [PMC SECURE] Drive CS1 pin with address bit 15 0 = DISABLED 1 = ENABLED
0	N1	EMEM_NUMDEV: [PMC SECURE] Number of populated DRAM devices. 0 = N1 1 = N2

20.11.2.6 EMC_REFCTRL_0

Refresh Control Register

The REFCTRL register allows software to enable or disable the refresh controller.

REF_VALID should be enabled after the initialization sequence is completed.

Boot requirements

- If per-device DPD is used, the DEVICE_REFRESH_DISABLE field should be parameterized in the BCT and written by the Boot ROM during cold boot.
- If per-device DPD is used, the DEVICE_REFRESH_DISABLE field should be parameterized the scratch registers and restored by the Boot ROM during warm boot.
- REF_VALID field should always be set to ENABLED for normal use, no need to parameterize in BCT/scratch.

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
31	DISABLED	REF_VALID: Enable refresh controller. 0 = DISABLED 1 = ENABLED
1:0	0x0	DEVICE_REFRESH_DISABLE: Disables refresh to individual attached device (1 bit per DRAM chip select).

20.11.2.7 EMC_PIN_0

Controls State of Selected DRAM Pins

The PIN register allows software to control the state of the selected external DRAM pins.

Boot requirements

- Both fields in this register should be set to NORMAL during cold boot.
- Both fields in this register should be set to NORMAL during warm boot.

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000100 (0bxxxxxxxxxxxxxxxxxxxxxxxx1xxx0xxx0)

Bit	Reset	Description
8	INACTIVE	PIN_RESET: Selects the level of the DDR3 RESET# pin. 0 = ACTIVE 1 = INACTIVE
4	NORMAL	PIN_DQM: Is used to always mask DRAM writes. This pin should only be used for initialization. Certain DRAM vendors require the DQM to be high during initialization. The register value should be set to NORMAL after the initialization sequence. 0 = NORMAL 1 = INACTIVE
0	POWERDOWN	PIN_CKE: Selects the level of the CKE pin. This can be used to place the DRAM in power down state. PIN_CKE value is applied all CKE pins. 0 = POWERDOWN 1 = NORMAL

20.11.2.8 EMC_TIMING_CONTROL_0

Triggers an Update of the Timing-Related Registers

The TIMING_CONTROL register is used by software to trigger parameter updates for timing parameter registers, RDQS/QUSE delay controls, and some DLL controls. Writing the TIMING_UPDATE field updates the active state of these registers with the programmed assembly state. The active state is updated during a safe interval determined by the EMC. If CLKCHANGE_REQ_ENABLE is enabled, the active value will automatically be updated on completion of the clock change.

> **Note:** Programming of this register does not trigger the shadow register update event immediately. To prevent shadow register programming issued after programming this register from being latched accidentally, always poll for TIMING_UPDATE_STALLED==0 after programming this register.

Boot Requirements

- Writing this register with 0x1 will trigger a timing update event, which should be used in both warm boot and cold boot sequences.

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TIMING_UPDATE

20.11.2.9 EMC_RC_0

DRAM Timing Parameter

This register is shadowed: see usage note at top.

Boot Requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2c | Read/Write: R/W | Reset: 0x0000007f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx1111111)

Bit	Reset	Description
6:0	0x7f	RC: [PMC] Specifies the row cycle time. This is the minimum number of cycles between activate commands to the same bank. DDR3: max(0, ceil(tRC/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1)) LPDDR2: ceil((tRASmin+tRPpb)/tCK) - 1

20.11.2.10 EMC_RFC_0

DRAM Timing Parameter

This register is shadowed: see usage note at top.

Boot Requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x30 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxx000111111)

Bit	Reset	Description
8:0	0x3f	RFC: [PMC] specifies the auto refresh cycle time. This is the minimum number of cycles between an auto refresh command and a subsequent auto refresh or activate command. DDR3: max(0, ceil(tRFC/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1)) LPDDR2: ceil(tRFCab/tCK) - 1

20.11.2.11 EMC_RAS_0

DRAM Timing Parameter

This register is shadowed: see usage note at top.

Boot Requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x34 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	RAS: [PMC] specifies the row active time. This is the minimum number of cycles between an activate command and a precharge command to the same bank. DDR3: max(0, ceil(tRASmin/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1)) LPDDR2: max(3, ceil(tRASmin/tCK)) - 1

20.11.2.12 EMC_RP_0

DRAM Timing Parameter

This register is shadowed: see usage note at top.

Boot Requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x38 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	RP: [PMC] specifies the row precharge time. This is the minimum number of cycles between a precharge command and an activate command to the same bank. DDR3: max(0, ceil(tRP/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1)) LPDDR2: max(3, ceil(tRP/tCK)) - 1

R2something and W2something registers are independent of burst length, and are relative to the last virtual CAS_ of a command. We start counting from the last data transfer as related to where CAS_ would be if BL = 4.

20.11.2.13 EMC_R2W_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x3c | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	R2W: [PMC] specifies the minimum number of cycles from any read command to any write command, irrespective of bank. This parameter guarantees the read->write turn-around time on the bus. DDR3: RL - WL + 2 - (CMD_2T_TIMING == 1 ? 1 : 0) + (CTT_TERMINATION == 1 ? max(0, CTT + CTT_DURATION - RDV - 2) : (EMC2PMACRO_CFG_XM2DQS_E_STRPULL_DQS ? 1 : 0)) + (EMC2PMACRO_CFG_QUSE_MODE == ALWAYS_ON ? 1 : 0) LPDDR2: RL - WL + 1 + ceil(tDQSCKmax/tCK) + (EMC2PMACRO_CFG_XM2DQS_E_STRPULL_DQS ? 1 : 0) + (EMC2PMACRO_CFG_QUSE_MODE == ALWAYS_ON ? 1 : 0) Largest programming value is 29

20.11.2.14 EMC_W2R_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x40 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	W2R: [PMC] specifies the minimum number of cycles from a write command to a read command, irrespective of bank. DDR3 : WL + max(4, ceil(tWTR/tCK)) - (CMD_2T_TIMING == 1 ? 1 : 0) LPDDR2 : WL + 1 + ceil(tWTR/tCK) Largest programming value is 29.

20.11.2.15 EMC_R2P_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x44 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	R2P: [PMC] specifies the minimum number of cycles from a read command to a precharge command for the same bank. DDR3 : max(4, ceil(tRTP/tCK)) - (CMD_2T_TIMING == 1 ? 2 : 1) LPDDR2 : BL/2 + max(2, ceil(tRTP/tCK)) - 1 - (is-lpddr2-s2 ? 1 : 2). Here is-lpddr2-s2 is 1 if the LPDDR2 is S2, otherwise (S4) it is 0

20.11.2.16 EMC_W2P_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x48 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	W2P: [PMC] Specifies the minimum number of cycles from a write command to a precharge command for the same bank. DDR3 : WL + max(2, ceil(tWR/tCK)) - (CMD_2T_TIMING == 1 ? 1 : 0) LPDDR2: WL + max(3, TWR)

20.11.2.17 EMC_RD_RCD_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x4c | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	RD_RCD: [PMC] Specifies the RAS to CAS delay. RD_RCD is the minimum number of cycles between an activate command and a read command to the same bank. DDR3 : ceil(tRCD/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1) LPDDR2: max(3, ceil(tRCD/tCK)) - 1

20.11.2.18 EMC_WR_RCD_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x50 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	WR_RCD: [PMC] Minimum number of cycles between an activate command and a write command to the same bank. DDR3 : ceil(tRCD/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1) LPDDR2: max(3, ceil(tRCD/tCK)) - 1

20.11.2.19 EMC_RRD_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x54 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3:0	0xf	RRD: [PMC] specifies the Bank X Act to Bank Y Act command delay. DDR3 : max(4, ceil(tRRD/tCK)) - (CMD_2T_TIMING == 1 ? 2 : 1) LPDDR2: max(2, ceil(tRRD/tCK)) - 1

20.11.2.20 EMC_REXT_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001)

Bit	Reset	Description
3:0	0x1	REXT: [PMC] specifies the read to read delay for reads when multiple physical devices are present. DDR3: USE_PER_DEVICE_DLY_TRIM_IB == 1 ? 2 : 1 LPDDR2: ceil(3/tCK + 0.5) + USE_PER_DEVICE_DLY_TRIM_IB ? 1 : 0

20.11.2.21 EMC_WDV_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	WDV: [PMC] the number of cycles to post (delay) write data from being asserted to the RAMs. DDR3 : WL - 1 LPDDR2 : WL 15 = MAX

20.11.2.22 EMC_QUSE_0

DRAM timing parameter

The QRST, QUSE, and RDV fields specify the delays from read to internal timing signals. These fields should be set as follows:

QUSE = CAS_LATENCY - 1 non-mobile SDRAM
= CAS_LATENCY - 2 for mobile SDRAM
QRST = CAS_LATENCY - 2
QSAFE >= RDV - QRST
RDV = CAS_LATENCY + 5
EINPUT = RDV - 7

Because SDRAM uses a tristating clock (the DQS), a method is needed to deal with the ambiguity of when DQS is tristated. Only one such method is supported: QUSE.

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000010)

Bit	Reset	Description
5:0	0x2	QUSE: [PMC] Tells the chip when to look for read return data. LPDDR2/DDR3: obtained from characterization

20.11.2.23 EMC_QRST_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000001)

Bit	Reset	Description
5:0	0x1	QRST: [PMC] time from expiration of QSAFE until reset is issued. DDR3 : CL - 2 LPDDR2 : RL - 2

20.11.2.24 EMC_QSAFE_0

DRAM timing parameter

When PERIODIC_QRST is enabled, the QSAFE parameter is intended to guarantee that the QRSTs will not interfere with pending reads (e.g. the queue is empty). This field must be set to at least (RDV - QRST).

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x68 | Read/Write: R/W | Reset: 0x00000007 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx00111)

Bit	Reset	Description
4:0	0x7	QSAFE: [PMC] Time from a read command to when it is safe to issue a QRST (delayed by the QRST parameter). LPDDR2/DDR3 QSAFE >= RDV - QRST

20.11.2.25 EMC_RDV_0

DRAM timing parameter

RDV is the read latency register. This register value is not negotiable; it will work with the right value and will not with the wrong value. It uses sequential timing, not combinational (i.e., maybe the silicon is fast enough type) timing.

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x6c | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx001000)

Bit	Reset	Description
5:0	0x8	RDV: [PMC] Time from read command to latching the read data from the pad macros. DDR3 : RL + 6 + f_ceil(fly-by-time/tCK) - (DRAM-DLL-is-off ? 1 : 0) LPDDR2: RL + 6 + ceil((tDQSCKmax + fly-by-time)/tCK) fly-by-time is board routing dependent, which usually should be less than 1.5ns DRAM-DLL-is-off is true below 150 MHz where DRAM DLL has to be turned off 35 = MAX

20.11.2.26 EMC_REFRESH_0

DRAM timing parameter

The value of REFRESH is calculated using the following formula:

```
{REFRESH, REFRESH_LO} = max[(tREF/#_of_rows) / (emc_clk_period) - 64,
                            (tREF/#_of_rows) / (emc_clk_period) * 97%]
```

For example, if the clock frequency is 133 MHz, and the refresh requirement is 64 ms per 4096 rows. The programming value is 0x7e3.

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x70 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxx0000000000011111)

Bit	Reset	Description
15:6	0x0	REFRESH: [PMC] specifies the interval between refresh requests.
5:0	0x1f	REFRESH_LO: [PMC]

20.11.2.27 EMC_BURST_REFRESH_NUM_0

DRAM timing parameter

BURST_REFRESH_NUM is used to specify the refresh burst count. The refresh controller will wait until BURST_REFRESH_NUM refreshes have been scheduled, then issue them all at once. This can result in a performance improvement in many cases.

Note: If tRAS(max) is less than the refresh interval (tREF/#_of_rows), tRAS(max) must be used instead of the refresh interval in the formula above. This is because refresh is used to satisfy tRAS(max) timing. Accordingly, BURST_REFRESH_NUM must be programmed in such a way that queuing up multiple refreshes does not violate tRAS(max) timing. Burst length = 2^BURST_REFRESH_NUM. Refreshes will be throttled to meet TREFBW limitation (8/window) if TREFBW > 0.

Note: Do not program this register to value non-zero unless for testing.

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	BR1	BURST_REFRESH_NUM: [PMC] Specify the refresh burst count. 0 = BR1 1 = BR2 2 = BR4 3 = BR8 4 = BR16 5 = BR32 6 = BR64 7 = BR128 8 = BR256 9 = BR512 9 = MAX

20.11.2.28 EMC_PDEX2WR_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x78 | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	PDEX2WR: Specify the timing delay from exit of powerdown mode to a write command. DDR3 : max(3, ceil(tXP/tCK)) - (CMD_2T_TIMING == 1 ? 2 : 1) LPDDR2 : ceil(tXP/tCK) - 1 Note: on DDR3, if slow power down exit mode is used and ODT is enabled, use max(10, ceil(tXPDLL/tCK)) instead of max(3, ceil(tXP/tCK)) in the formula Largest allowed value is 62

20.11.2.29 EMC_PDEX2RD_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x7c | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	PDEX2RD: Specify the timing delay from exit of powerdown mode to a read command. DDR3 : (use-slow-pd-exit-mode ? max(10, ceil(tXPDLL/tCK)) : max(3, ceil(tXP/tCK))) - (CMD_2T_TIMING == 1 ? 2 : 1) LPDDR2 : ceil(tXP/tCK) - 1 use-slow-pd-exit-mode is 1 if slow power down exit mode is used (MR0 bit 12 is 0). Largest allowed value is 62

20.11.2.30 EMC_PCHG2PDEN_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x80 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	PCHG2PDEN: [PMC] Specify the timing delay from a precharge command to powerdown entry. DDR3 : 1 LPDDR2 : f_ceil(tRP]/tCK) - (CMD_2T_TIMING == 1 ? 1 : 0)

20.11.2.31 EMC_ACT2PDEN_0

[PMC] DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x84 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	ACT2PDEN: Specify the timing delay from an activate, MRS or EMRS command to power-down entry. DDR3/LPDDR2: 0

20.11.2.32 EMC_AR2PDEN_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x88 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxx000011111)

Bit	Reset	Description
8:0	0x1f	AR2PDEN: [PMC] Specify the timing delay from an autorefresh command to powerdown entry. DDR3 : max(7, ceil((tXPDLL - tXP)/tCK)) LPDDR2 : 1

20.11.2.33 EMC_RW2PDEN_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x8c | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx011111)

Bit	Reset	Description
5:0	0x1f	RW2PDEN: [PMC] Specify the timing delay from a read/write command to powerdown entry. Auto-precharge timing must be taken into account when programming this field DDR3: max(RL+5, WL+4+TWR) - 1 LPDDR2: max(RL+BL/2+ceil((tDQSCKmax+1)/tCK)+1, WL+BL/2+1+ceil((tWR+1)/tCCK)) - 1

20.11.2.34 EMC_TXSR_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x90 | Read/Write: R/W | Reset: 0x000003fe (0bxxxxxxxxxxxxxxxxxxxxxxxx1111111110)

Bit	Reset	Description
9:0	0x3fe	TXSR: [PMC] Cycles between self-refresh exit & first DRAM command that does not require a locked DLL. Largest allowed value is 0x3fe DDR3: max(5, ceil(tXSR/tCK)) LPDDR2: max(2, ceil(tXSR/tCK))

20.11.2.35 EMC_TCKE_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x94 | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	TCKE: Specify the minimum CKE high pulse width. DDR3: Use-slow-pd-exit-mode ? max(10, ceil(max(tCKESR, uXPDLL)/tCK)) : max(4, ceil(max(tXP,tXP)/tCK)) LPDDR2: max(3, tCKE, ceil(tXP/tCK)) use-slow-pd-exit-mode is 1 if slow power down exit mode is used (MR0 bit 12 is 0

20.11.2.36 EMC_TFAW_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	TFAW: [PMC] specify the width of the FAW (four-activate window) for 8-bank devices. Set to 0 to disable this timing check. Only 4 activates may occur within the rolling window. . DDR3/LPDDR2: ceil(tFAW/tCK)

20.11.2.37 EMC_TRPAB_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:0	0x0	TRPAB: [PMC] Specify precharge-all tRP allowance for 8-bank devices. Setting this field to 0 will cause EMC to use TRP.TRP for precharge-all. DDR3 : ceil(tRP/tCK) + 1 LPDDR2: max(3, ceil((tRPpb+3)/tCK

20.11.2.38 EMC_TCLKSTABLE_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xa0 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx01111)

Bit	Reset	Description
4:0	0xf	TCLKSTABLE: [PMC] Specify minimum number of cycles of a stable clock period prior to exiting powerdown or self-refresh modes. DDR3 : max(5, ceil(tCKSRX/tCK)) - 1 LPDDR2: 2

20.11.2.39 EMC_TCLKSTOP_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xa4 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx01111)

Bit	Reset	Description
4:0	0xf	TCLKSTOP: [PMC] Delay from last command to stopping the external clock to DRAM devices.

Bit	Reset	Description
		DDR3: max(5, ceil(tCKSRE/tCK)) - 1 LPDDR2: 2

20.11.2.40 EMC_TREFBW_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13:0	0x0	TREFBW: [PMC] specify the width of the burst-refresh window. If set to a non-zero value, only 8 refreshes will occur in this rolling window. Set to 0 to disable this timing check. DDR3: ceil(tREFI/tCK) LPDDR2: ceil(tREFBW/tCK)

20.11.2.41 EMC_QUSE_EXTRA_0

QUSE_EXTRA is combined with QUSE to add extra cycles to data return window. This is for use with DQS pulldowns in LPDDR2 where start of read preamble could vary by more than 1 cycle.

Example: To extend end of QUSE window by 1 cycle, set QUSE_EXTRA = QUSE + 1. To start the QUSE window earlier by 1 cycle, set QUSE_EXTRA = QUSE - 1. Disabled if QUSE_EXTRA = 0.

QUSE_PULLED_MORE_THAN_BL: The QUSE pulse width is equal to BL and can be extended with QUSE_EXTRA. If QUSE is pulled in more than the BL, this field needs to be set to 1.

QUSE_EXTENDED_MORE_THAN_BL: If QUSE is extended by more than the BL at the end, this field needs to be set to 1.

Only one of the QUSE_PULLED_MORE_THAN_BL or QUSE_EXTENDED_MORE_THAN_BL can be set 1 at any time because the QUSE window can be either pulled in or extended for any setting.

By default QUSE_PULLED_MORE_THAN_BL/QUSE_EXTENDED_MORE_THAN_BL should be set to 0.

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00xx000000)

Bit	Reset	Description
9	0x0	QUSE_EXTENDED_MORE_THAN_BL: [PMC]
8	0x0	QUSE_PULLED_MORE_THAN_BL: [PMC]
5:0	0x0	QUSE_EXTRA: [PMC2] DDR3/LPDDR2: (EMC2TMC_CFG_XM2DQS_E_SCHMT == 1) ? (QUSE + 1) : 0

20.11.2.42 EMC_ODT_WRITE_0

For DDR3, ODT may be enabled during writes and disabled during reads via control of ODT pin.

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000020 (0b00xxxxxxxxxxxxxxxxxx0000xx10x000

Bit	Reset	Description
31	0x0	ENABLE_ODT_DURING_WRITE: enables ODT to be turned on prior to issuing write to DRAM. [PMC] If ENABLE_ODT_DURING_WRITE = 1 and DISABLE_ODT_DURING_READ = 0, ODT will always be enabled after 1st write.
30	0x0	ODT_B4_WRITE: [PMC] If this field == 1, ODT is turned on ODT_WR_DELAY cycles prior to DRAM WRITE command. If this field == 0, ODT is turned on ODT_WR_DELAY cycles after DRAM WRITE command. Set ODT_B4_WRITE to 1 if (WL - ceiling(tAOND) - 2) < 0. This field is not used for DDR3.
11:8	0x0	ODT_WR_DURATION: [PMC] for LPDDR3, indicate how long to assert ODT by.
5	0x1	SHARE_ONE_ODT: [PMC] For LPDDR3 only. When enabled, ODT[1] is not used. The ODT for both ranks will be sharing the same ODT[0]. When disabled, ODT[0] is controlling rank0, ODT[1] is controlling rank1.
4	0x0	DRIVE_BOTH_ODT: [PMC] If this field = 0, only the ODT for the requested device will be asserted during write. If this field = 1, ODTs for both devices will be asserted during write.
2:0	0x0	ODT_WR_DELAY: [PMC] Set this field = ABS (WL - ceiling(tAOND) - 2). The valid programming range is 0 <= ODT_WR_DELAY <= 2, if ODT_B4_WRITE=0, 0 <= ODT_WR_DELAY <= 1, if ODT_B4_WRITE=1. For DDR3, this field = (WL - 3). For LPDDR3, this field = (WL - 2).

20.11.2.43 EMC_ODT_READ_0

This register is not used. Please keep it in reset value.

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
31	0x0	DISABLE_ODT_DURING_READ: enables ODT to be turned off prior to issuing read to DRAM. If this field == 0, ODT state will not be changed for reads. If this field == 1, Turn off ODT prior to READ command (has no effect if ODT ENABLE_ODT_DURING_WRITE == 0, as ODT will always be disabled). This field is not used for DDR3.
30	0x0	ODT_B4_READ: If this field == 1, ODT is turned off ODT_RD_DELAY cycles prior to DRAM READ command. If this field == 0, ODT is turned off ODT_RD_DELAY cycles after DRAM READ command. Set ODT_B4_READ to 1 if (RL - ceiling(tAOFD) - 2) < 0. This field is not used for DDR3.
2:0	0x0	ODT_RD_DELAY: Set this field = ABS (RL - ceiling(tAOFD) - 2). The valid programming range is 0 <= ODT_RD_DELAY <= 2 if ODT_B4_READ=0, 0 <= ODT_RD_DELAY <= 1 if ODT_B4_READ=1 This field is not used for DDR3.

20.11.2.44 EMC_WEXT_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xb8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	WEXT: [PMC] specifies the write to write delay for writes when multiple physical devices are present. DDR3: max(0, (USE_PER_DEVICE_DLY_TRIM_OB == 1 ? 5 : 0) - (CMD_2T_TIMING == 1 ? 1 : 0)) LPDDR2: USE_PER_DEVICE_DLY_TRIM_OB == 1 ? 5 : 0

20.11.2.45 EMC_CTT_0

DRAM timing parameter

CTT controls timing of internal read terminations. Based on read latency (CTT must frame read data). Adding non-RDV linked timing control for flexibility. Has no effect unless CTT_TERMINATION == ENABLED || E_PUTERM_DQ == ENABLED It is sequential timing, not combinational (i.e., maybe the silicon is fast enough type) timing.

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xbc | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx001000)

Bit	Reset	Description
5:0	0x8	CTT: [PMC] time from read command to CTT turn-on on DDR3 ? (CTT_TERMINATION == 1) ? CL + 7 : 0 LPDDR2: 0 35 = MAX

20.11.2.46 EMC_RFC_SLR_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8:0	0x0	RFC_SLR: [PMC] Specifies the stagger refresh cycle time between ranks. This is the minimum number of cycles between an auto refresh command to one rank and a subsequent auto refresh to another rank. A zero value indicates stagger refresh is disabled.

20.11.2.47 EMC_MRS_WAIT_CNT2_0

This register allows the delay between an MRS/EMRS/MRW/MRR command and the following DRAM command

Boot Requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xc4 | Read/Write: R/W | Reset: 0x0208000f (0bxxxxxx1000001000xxxxxx0000001111) | Default: 0x03ff03ff

Bit	Reset	SW Default	Description
25:16	0x208	0x3ff	MRS_EXT2_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending an MRS ext2 command.
9:0	0xf	0x3ff	MRS_EXT1_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending an MRS ext1 command.

20.11.2.48 EMC_MRS_WAIT_CNT_0

This register allows the delay between an MRS/EMRS/MRW/MRR command and the following DRAM command.

Boot Requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xc8 | Read/Write: R/W | Reset: 0x0208000f (0bxxxxxx1000001000xxxxxx0000001111)

Bit	Reset	SW Default	Description
25:16	0x208	0x3ff	MRS_LONG_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending an MRS long command.
9:0	0xf	0x3ff	MRS_SHORT_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending an MRS short command.

20.11.2.49 EMC_MRS_0

Command Trigger: MRS

The MRS register allows software to issue an MRS command.

BA0, BA1 are used to address MRS or EMRS registers in DRAM. Although this register can also program EMRS, use the EMRS register so that the hardware registers can shadow what is in the DRAM.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xxxx00xxxxxx00000000000000)

Bit	Reset	Description
31:30	0x0	MRS_DEV_SELECTN: [PMC] Active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
27	0x0	USE_MRS_EXT_CNT: [PMC] 0 =USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1=USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_MRS_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	MRS_BA: [PMC] Set to 0x0 for MRS.
13:0	0x0	MRS_ADR: [PMC] Mode-register data to be written.

20.11.2.50 EMC_EMRS_0

Command trigger: EMRS

The EMRS register allows software to issue an EMRS command.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xd0 | Read/Write: R/W | Reset: 0x00000000 (0b00xxx0xxxx00xxxxxx00000000000000)

Bit	Reset	Description
31:30	0x0	EMRS_DEV_SELECTN: [PMC] Active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
27	0x0	USE_EMRS_EXT_CNT: [PMC] 0 = USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1 = USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_EMRS_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	EMRS_BA: [PMC] Set to 0x1 for EMRS (and where applicable, 0x2 for EMRS2, and 0x3 for EMRS3).
13:0	0x0	EMRS_ADR: [PMC] Mode-register data to be written.

20.11.2.51 EMC_REF_0

Command Trigger: Refresh

Note: The REF register allows software to issue refresh commands. This is done to ensure proper DRAM initialization.
REF_NUM number of refreshes are issued to both devices regardless of the REF_DEV_SELECTN setting.

Boot requirements

- This register triggers a refresh command. REF_CMD should be written with 0x1 during cold boot to trigger refresh commands during DRAM initialization.

Offset: 0xd4 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxx00000000xxxxxx00)

Bit	Reset	Description
31:30	0x0	REF_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
15:8	0x0	REF_NUM: perform (REF_NUM + 1) refresh cycles.
1	0x0	REF_NORMAL: 0 = execute first refresh immediately (this is use during DRAM initialization). 1 = execute refresh through normal refresh mechanism (this should be used in normal operation).
0	0x0	REF_CMD: causes the hardware to perform a REFRESH to all DRAM banks.

20.11.2.52 EMC_PRE_0

Command Trigger: Precharge-All

The PRE register allows software to issue a precharge all command. This command may be used to ensure proper DRAM initialization.

Boot requirements

- This register triggers a precharge-all command. PRE_CMD should be written with 0x1 during cold boot to trigger refresh commands during DRAM initialization.
- If per-device DPD is used, the PRE_DEV_SELECTN field should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:30	0x0	PRE_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
0	0x0	PRE_CMD: causes the hardware to perform a PRECHARGE to all DRAM banks.

20.11.2.53 EMC_NOP_0

Command Trigger: NOP

The NOP register allows software to issue an explicit NOP command. This command may be used to ensure proper DRAM initialization.

Boot requirements

- This register triggers a no-operation command. NOP_CMD should be written with 0x1 during cold boot to trigger no-op commands during DRAM initialization.

Offset: 0xdc | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
31:30	0x0	NOP_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
0	0x0	NOP_CMD: causes the hardware to perform a NOP to all DRAM banks.

20.11.2.54 EMC_SELF_REF_0

Command Trigger: SELF REFRESH

The SELF_REF register allows software to issue self-refresh commands.

Do not program this register when a clock change sequence is on-going. Check the CLKCHANGE_COMPLETE_INT value if not sure.

Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:30	0x0	SREF_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1, 0x3 for neither device.

Bit	Reset	Description
0	DISABLED	SELF_REF_CMD: causes the hardware to issue a SELF_REFRESH command. While CMD:ENABLED, the CKE pin is held deasserted. The CMD:ENABLED state will override the PIN:CKE setting. The DRAM will ignore all accesses until CMD:DISABLED. 0 = DISABLED 1 = ENABLED

20.11.2.55 EMC_DPD_0

Command Trigger: Deep Power Down

The DPD register allows software to issue a deep power down command.

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:30	0x0	DPD_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
0	DISABLED	DPD_CMD: causes the hardware to issue the deep power down command (Burst Terminate with cke low). While in DPD mode, the DRAM will not maintain data integrity. While CMD:ENABLED, the CKE pin is held deasserted. The CMD:ENABLED state will override the PIN:CKE setting. The DRAM will ignore all accesses until CMD:DISABLED. 0 = DISABLED 1 = ENABLED

20.11.2.56 EMC_MRW_0

Command Trigger: MRW

Mode Register Write: LPDDR2-only version of MRS/EMRS

Boot requirements

- This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0xe8 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000

Bit	Reset	Description
31:30	0x0	MRW_DEV_SELECTN: [PMC] Active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW_EXT_CNT: [PMC] 0 = USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1 = USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_MRW_LONG_CNT: [PMC] Indicate to use long or short MRS wait count.
23:16	0x0	MRW_MA: [PMC] Register address
7:0	0x0	MRW_OP: [PMC] Data to be written

20.11.2.57 EMC_MRR_0

Command Trigger: MRR

Mode Register Read: LPDDR2 only

Sequence

1. Read MRR until EMC_STATUS.MRR_DIVLD=0 (ok to skip if it is sure there are no pending MRR reads)

2. Write this register with the desired addr (MA) and device (DEV_SELECTN), device needs to be either DEV0 or DEV1: writing to both is illegal

3. Poll EMC_STATUS.MRR_DIVLD=1, or if using interrupt, wait for MRR_DIVLD_INT

4. Read back MRR. The value is in MRR_DATA field.

Note: It is okay to issue new MRR requests while there are on-going requests. e.g., to issue 3 MRR requests, follow these steps: 1,2,2,2,3,4,3,4,3,4. Data read back in step 4 is in the same order requested in step 2.

To make sure the EMC is available for new MRR requests, poll for EMC_STATUS.MRR_FIFO_SPACE > 0 before step 2.

If using 2 x16 DRAM, MRR_DATA[15:8] can be used to store MRR from 2nd DRAM on same CS (configured via CFG_2.MRR_BYTESEL_X16)

Offset: 0xec | Read/Write: R/W | Reset: 0x0000XXXX (0b00xx00xx00000000xxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31:30	RW	0x0	MRR_DEV_SELECTN: active-low chip-select, choose which device to send the command to. (enum for safety). 0 = ILLEGAL 1 = DEV1 2 = DEV0 3 = RESERVED
27	RW	0x0	USE_MRR_EXT_CNT: 0 = USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1 = USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	RW	SHORT	USE_MRR_LONG_CNT: Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	RW	0x0	MRR_MA: register address
15:0	RO	X	MRR_DATA: data returned

20.11.2.58 EMC_XM2DQSPADCTRL3_0

Offset: 0xf8 | Read/Write: R/W | Reset: 0x20820800 (0bx01000x01000x01000x01000xx0xxxx0)

Bit	Reset	Description
30:26	0x8	EMC2PMACRO_CFG_XM2DQS_BYTE3_VREF_DQS: [PMC]
24:20	0x8	EMC2PMACRO_CFG_XM2DQS_BYTE2_VREF_DQS: [PMC]
18:14	0x8	EMC2PMACRO_CFG_XM2DQS_BYTE1_VREF_DQS: [PMC]
12:8	0x8	EMC2PMACRO_CFG_XM2DQS_BYTE0_VREF_DQS: [PMC]
5	0x0	EMC2PMACRO_CFG_XM2DQS_E_VREF_DQS: [PMC]
0	0x0	EMC2PMACRO_CFG_XM2DQS_E_STRPULL_DQS: [PMC]

20.11.2.59 EMC_FBIO_CFG5_0

FBIO Configuration Register

The trimmer value may be overridden by setting using MULT==0 and OFFS = value << 4

The FBIO_CFG5 register controls the FBIO I/O cells.

The following fields are shadowed: DIFFERENTIAL_DQS, CTT_TERMINATION, DQS_PULLD, CMD_2T_TIMING.

Writes to these fields will not take effect until the active value is updated via TIMING_UPDATE or (if enabled) CLKCHANGE_REQ.

Boot requirements

- This register (except for field DISABLE_CONCURRENT_AUTOPRE) should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register (except for field DISABLE_CONCURRENT_AUTOPRE) should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x104 | Read/Write: R/W | Reset: 0x0000a801 (0bxxxxxxxxxxxxxxxx01010100000xx0001)

Bit	Reset	Description
16	DISABLED	LPDDRX_ADR_PIN_SWIZZLE_EN: [PMC] Enable LPDDRx pin swizzling A14->A0, RAS->A1, CAS->A2, WE->A3 0 = DISABLED 1 = ENABLED
15:13	DIRECT_QUSE	EMC2PMACRO_CFG_QUSE_MODE: [PMC] Select QUSE mode (NORMAL = off-chip) (ALWAYS_ON = requires DQS PULL) (INTERNAL_LPBK selects on-chip loopback). (DIRECT_QUSE = directly controlled QUSE) 0 = NORMAL 1 = ALWAYS_ON 2 = INTERNAL_LPBK 3 = PULSE_INT 4 = PULSE_EXT 5 = DIRECT_QUSE 6 = RESERVED
12	DISABLED	CMD_2T_TIMING: [PMC] Enable 2T command timing (RAS/CAS/WE/ROW/A/BA). 0 = DISABLED 1 = ENABLED
11	ENABLED	CFG_QUSE_EXTEND_HALF_CLK: [PMC] Extend the QUSE window by half clk in DIRECT_QUSE mode 0 = DISABLED 1 = ENABLED
10	DISABLED	DISABLE_CONCURRENT_AUTOPRE: [PMC] Disables reads/writes to a device until the precharge command has been issued by the DRAM internally. 0 = DISABLED 1 = ENABLED
9	DISABLED	DQS_PULLD: [PMC] Enables pull-downs on DQS lines (and pull-ups on DQS_N if DIFFERENTIAL_DQS). 0 = DISABLED 1 = ENABLED
8	DISABLED	CTT_TERMINATION: [PMC] Enables CTT_TERMINATION mode in pads (DDR2 support) 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
7	DISABLED	DIFFERENTIAL_DQS: [PMC] Enables differential signaling on DQS strobes (LPDDR2/DDR2 options) 0 = DISABLED 1 = ENABLED
6	DISABLED	CFG_E_PUTERM_DQ: [PMC] Enables E_PUTERM_DQ mode in pads (DDR3 support). In the IOBRICK, the E_PUTERM_DQ mode overrides CTT_TERMINATION mode. 0 = DISABLED 1 = ENABLED
3:2	BURST4	DRAM_BURST: [PMC] Specifies the burst length to use for the attached device(s). ON_THE_FLY BC4/BL8 is only use for DDR3. 0 = BURST4 1 = BURST8 2 = ON_THE_FLY 3 = RESERVED
1:0	DDR1	DRAM_TYPE: [PMC] Specifies which DRAM protocol to use for the attached device(s). 0 = DDR3 1 = DDR1 2 = LPDDR2 3 = DDR2

20.11.2.60 EMC_FBIO_CFG6_0

FBIO Configuration Register

QUSE_LATE determines how much added delay FBIO should add to the QUSE path. QUSE needs to align (approximately) with the incoming DQS in order to qualify it, since the incoming DQS/feedback clock is not always valid.

DRAMC can position QUSE with m2clk granularity (2 bit times). QUSE_LATE provides finer granularity of 1/2 an M2CLK cycle (1/2 bit time). The amount of delay added is primarily a function of the round trip wire delay to/from the DRAM. Other portions of the delay (driver and receiver delay) are compensated for by delay through a non-bonded QUSE pad cell.

Additional delay can be added to QUSE vi XFORM_QUSEx_MULT and XFORM_QUSEx_OFFS (applied to DLL offset).

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x114 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx010)

Bit	Reset	Description
2:0	0x2	CFG_QUSE_LATE: [PMC]

20.11.2.61 EMC_ACPD_CONTROL_0

Threshold for ACPD

Offset: 0x124 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	ACPD_THRESHOLD: [PMC] number of idle cycles to wait before allowing power-down entry

20.11.2.62 EMC_EMRS2_0

Command Trigger: EMRS2

The EMRS2 register allows software to issue an EMRS2 command.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x12c | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xxxx00xxxxxx00000000000000)

Bit	Reset	Description
31:30	0x0	EMRS2_DEV_SELECTN: [PMC] active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
27	0x0	USE_EMRS2_EXT_CNT: [PMC] 0=>USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1=>USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_EMRS2_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	EMRS2_BA: [PMC] Set to 0x1 for EMRS (and where applicable, 0x2 for EMRS2, and 0x3 for EMRS3).
13:0	0x0	EMRS2_ADR: [PMC] mode-register data to be written.

20.11.2.63 EMC_EMRS3_0

Command Trigger: EMRS2

The EMRS3 register allows software to issue an EMRS3 command.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x130 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xxxx00xxxxxx00000000000000)

Bit	Reset	Description
31:30	0x0	EMRS3_DEV_SELECTN: [PMC] active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
27	0x0	USE_EMRS3_EXT_CNT: [PMC] 0=>USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1=>USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_EMRS3_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	EMRS3_BA: [PMC] Set to 0x1 for EMRS (and where applicable, 0x2 for EMRS2, and 0x3 for EMRS3).
13:0	0x0	EMRS3_ADR: [PMC] mode-register data to be written.

20.11.2.64 EMC_MRW2_0

Command Trigger: MRW2

Mode Register Write: LPDDR2-only version of MRS/EMRS

Boot requirements

This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x134 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x0	MRW2_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW2_EXT_CNT: [PMC] 0=>USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1=>USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_MRW2_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	0x0	MRW2_MA: [PMC] register address
7:0	0x0	MRW2_OP: [PMC] data to be written

20.11.2.65 EMC_MRW3_0

Command Trigger: MRW3

Mode Register Write: LPDDR2-only version of MRS/EMRS

Boot requirements

This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x138 | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x0	MRW3_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW3_EXT_CNT: [PMC] 0=>USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1=>USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_MRW3_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	0x0	MRW3_MA: [PMC] register address
7:0	0x0	MRW3_OP: [PMC] data to be written

20.11.2.66 EMC_MRW4_0

Command Trigger: MRW4

Mode Register Write: LPDDR2-only version of MRS/EMRS

Boot requirements

This register triggers a mode register write command. Multiple mode-register writes may be required by the cold boot sequence. Such commands should be parameterized in the BCT and written by the Boot ROM during cold boot.

Offset: 0x13c | Read/Write: R/W | Reset: 0x00000000 (0b00xx00xx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x0	MRW4_DEV_SELECTN: [PMC] active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
27	0x0	USE_MRW4_EXT_CNT: [PMC] 0=>USE_MRS_LONG_CNT will select between SHORT(0)/LONG(1), 1=>USE_MRS_LONG_CNT will select between EXT1(0)/EXT2(1).
26	SHORT	USE_MRW4_LONG_CNT: [PMC] Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
23:16	0x0	MRW4_MA: [PMC] register address
7:0	0x0	MRW4_OP: [PMC] data to be written

20.11.2.67 EMC_EINPUT_0

DRAM Timing Parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:0	0x0	EINPUT: [PMC] specifies when to assert EINPUT for a read, should normally be the same as QUSE.

20.11.2.68 EMC_EINPUT_DURATION_0

DRAM Timing Parameter

The minimum for EINPUT_DURATION is 4.

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x150 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx100)

Bit	Reset	Description
2:0	0x4	EINPUT_DURATION: [PMC] Specifies how long the EINPUT should be asserted.

20.11.2.69 EMC_PUTERM_EXTRA_0

DRAM Timing Parameter

PUTERM_EXTRA is used only when the DIRECT_QUSE mode is enabled to enable the dynamic puterm before the QUSE window.

PUTERM_PULLED_MORE_THAN_BL: The PUTERM pulse width is equal to BL+1 and can be extended with PUTERM_EXTRA. If PUTERM is pulled in more than BL, this field needs to be set to 1.

PUTERM_EXTENDED_MORE_THAN_BL: If PUTERM is extended by more than BL at the end, this field needs to be set to 1. Only one of the PUTERM_PULLED_MORE_THAN_BL or PUTERM_EXTENDED_MORE_THAN_BL can be set to 1 at any time because the PUTERM window can be either pulled in or extended for any setting. By default PUTERM_PULLED_MORE_THAN_BL/PUTERM_EXTENDED_MORE_THAN_BL should be set to 0.

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x154 | Read/Write: R/W | Reset: 0x00010000 (0bxxxxxxxxxx000001xxxxxx00xx000000)

Bit	Reset	Description
21:16	0x1	PUTERM: [PMC] tells the chip when to assert dynamic PUTERM for read return data.
9	0x0	PUTERM_EXTENDED_MORE_THAN_BL: [PMC]
8	0x0	PUTERM_PULLED_MORE_THAN_BL: [PMC]
5:0	0x0	PUTERM_EXTRA: [PMC]

20.11.2.70 EMC_TCKESR_0

DRAM Timing Parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x158 | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	TCKESR: [PMC] Specify minimum low CKE pulse width for self-refresh mode.

20.11.2.71 EMC_TPD_0

DRAM Timing Parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x15c | Read/Write: R/W | Reset: 0x0000003e (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111110)

Bit	Reset	Description
5:0	0x3e	TPD: [PMC] specify minimum low CKE pulse width for power-down mode.

20.11.2.72 EMC_AUTO_CAL_CONFIG_0

Auto-Calibration Settings for EMC Pads

Boot requirements

- This register (except for AUTOCAL_SLW_OVERRIDE and AUTO_CAL_START(trigger)) should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register (except for AUTOCAL_SLW_OVERRIDE and AUTO_CAL_START(trigger)) should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2a4 | Read/Write: R/W | Reset: 0x00f10000 (0b00000x001111x001xxx00000xxx00000

Bit	R/W	Reset	Description
31	RO	0x0	AUTO_CAL_START: [PMC] Writing a one to this bit starts the calibration state machine. This bit must be set even if the override is set in order to latch in the override value.
30	RW	0x0	AUTO_CAL_OVERRIDE: [PMC] 0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting 1 (override) : use AUTO_CAL_PU/PD_OFFSET register values directly
29	RW	DISABLED	AUTO_CAL_ENABLE: [PMC] 1 (normal operation): use EMC generated pullup/down (override or autocal) 0 (disabled): use emc2tmc_cfg* register settings for pullup/down 0 = DISABLED 1 = ENABLED
28	RW	0x0	AUTOCAL_SLW_OVERRIDE: [PMC] 0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output DRDVDN/UP_SLWR/F[3:0] = AUTO_CAL_PULLDOWN/UP[4:1] 1 (override) use EMC2TMC_CFG*_DRVDN/UP_SLWR/F pins to control pad slew inputs
27	RW	0x0	AUTO_CAL_SLAVE_MODE: [PMC]. 0 master mode (used by channel connected to the comp pads). 1 slave mode (channel uses the PU/PD codes determined by the master)
25:20	RW	0xf	AUTO_CAL_E_CAL_UPDATE: [PMC] number of EMC clocks E_CAL_UPDATE is asserted
18:16	RW	0x1	AUTO_CAL_STEP: [PMC] calibration step interval (in microseconds)
12:8	RW	0x0	AUTO_CAL_PD_OFFSET: [PMC] 2's complement offset for the pull-down value
4:0	RW	0x0	AUTO_CAL_PU_OFFSET: [PMC] 2's complement offset for the DQ/DQS pull-up value

20.11.2.73 EMC_AUTO_CAL_INTERVAL_0

EMC Pad Calibration Interval

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
20:0	0x0	AUTO_CAL_INTERVAL: [PMC] 0: Calibrate once. Otherwise, auto-calibration occurs at intervals equivalent to the programmed number of microseconds.

20.11.2.74 EMC_AUTO_CAL_STATUS_0

EMC Pad Calibration Status

Offset: 0x2ac | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTO_CAL_ACTIVE: One when auto calibrate is active - valid only after auto calibrate sequence has completed (EMC_CAL_ACTIVE == 0)
28:24	X	AUTO_CAL_PULLDOWN_ADJ: Pull-down code sent to pads
20:16	X	AUTO_CAL_PULLUP_ADJ: Pull-up code sent to pads
12:8	X	AUTO_CAL_PULLDOWN: Pull-down code generated by auto-calibration
4:0	X	AUTO_CAL_PULLUP: Pull-up code generated by auto-calibration

20.11.2.75 EMC_REQ_CTRL_0

Request Status/Control

When either STALL_ALL_READS and/or STALL_ALL_WRITES is asserted, the stalling of read and/or write requests will not take effect until the status bit from EMC_STATUS register's NO_OUTSTANDING_TRANSACTIONS field is asserted.

Offset: 0x2b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	STALL_ALL_WRITES: Stall incoming write transactions
0	0x0	STALL_ALL_READS: Stall incoming read transactions

20.11.2.76 EMC_EMC_STATUS_0

EMC State-Machine Status

DRAM_IN_POWERDOWN, DRAM_IN_SELF_REFRESH, DRAM_IN_DPD: active high signal indicating current DRAM status for each of these modes, with 1 status bit per device, bit[0] = dev0 status, bit[1] = dev1 status.

Example: DRAM_IN_SELF_REFRESH = 0x3, both devices are in self-refresh, DRAM_IN_DPD= 0x2 would indicate only dev1 is in deep-power-down mode.

> **Note:** If EMC is reset or powered down, the actual DRAM state could be different than indicated by these status bits. These bits do not reflect manually entered/exited powerdown or self-refresh (via use of PIN_CKE).

Offset: 0x2b4 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
27:26	X	ACPD_FSM_IDLE: Indicate dev[n] power-down FSM is idle.
25:24	X	DSR_FSM_IDLE: Indicate dev[n] dynamic self refresh FSM is idle.
23	X	TIMING_UPDATE_STALLED: Indicates timing update triggered by TIMING_CONTROL.TIMING_UPDATE is not yet completed.
22	X	ZQ_FSM_IDLE: Indicates auto ZQ state machine is idle.
21	X	CFG_ZQ_ACTIVE: Indicates ZQ configuration access is active.
20	X	MRR_DIVLD: MRR data available for reading
19:16	X	MRR_FIFO_SPACE: MRR FIFO space available

Bit	Reset	Description
13:12	X	DRAM_IN_DPD: dev[n] has been put into deep powerdown state
9:8	X	DRAM_IN_SELF_REFRESH: dev[n] has been put into self-refresh (will remain until SR exit cmd).
5:4	X	DRAM_IN_POWERDOWN: dev[n] has entered powerdown state (incoming req's will awaken if not stalled)
2	X	NO_OUTSTANDING_TRANSACTIONS: All non-stalled requests have completed
0	X	EMC_REQ_FIFO_EMPTY: Request FIFO is empty

20.11.2.77 EMC_CFG_2_0

EMC Configuration

Clock change sequencing: Once the divider is reprogrammed, CAR signals to EMC that a clock change is pending. If enabled, EMC stalls incoming requests, drains outstanding requests, and, if CLKCHANGE_(PD|SR)_ENABLE is enabled, puts DRAM into power-down (or self-refresh) before signalling to CAR that it is idle and ready for the change to happen. CAR will then change the divider/pll reprogramming. Once complete, EMC updates its shadow registers (assuming they may have been reprogrammed for new clock setting), unstalls requests, and resumes operation with new clock settings.

Boot requirements

- This register (except for fields DRAMC_PRE_B4_ACT, MRR_BYTESEL_X16, MRR_BYTESEL) should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register (except for fields DRAMC_PRE_B4_ACT, MRR_BYTESEL_X16, MRR_BYTESEL) should be saved in the scratch registers and restored by the Boot ROM during warm boot.
- If the OS needs the MRR_BYTESEL* fields set to non-default values to perform a mode-register read, it needs to correctly program these values before performing the MRR.

Offset: 0x2b8 | Read/Write: R/W | Reset: 0x00008047 (0b000000000000xx001000000001xxx111

Bit	Reset	Description
31	DISABLED	DRAMC_PRE_B4_ACT: [PMC] Reserved bit, gives priority to activates over precharges, determining which (precharge/activate) is processed first if both are pending and unblocked. 0 = DISABLED 1 = ENABLED
13	DISABLED	USE_PER_DEVICE_DLY_TRIM_OB: [PMC] Reserved bit. Allows using different outbound delay trim values per device. When enabled, WDV must be programmed to >= 2 and WEXT >=5. 0 = DISABLED 1 = ENABLED
12	DISABLED	USE_PER_DEVICE_DLY_TRIM_IB: [PMC] Reserved bit. Allows using different inbound delay trim values per device. When enabled, REXT/RDV/QUSE must be programmed to >= 2. 0 = DISABLED 1 = ENABLED
11	0x0	DIS_CNTR_WITH_CFG_TIMING_UPDATE: [PMC] Reserved bit. 1 = Disable reset of timing parameter counters with TIMING_CONTROL.TIMING_UPDATE.
9:8	LPDDR2	PIN_CONFIG: [PMC] Remaps address/command pins for LPDDR2_POP ball-out otherwise uses standard LPDDR2 pin configuration. 0 = LPDDR2 1 = LPDDR2_POP 2 = RESERVED
7	0x0	EARLY_TRFC_8_CLK: [PMC] Reserved bit. 1 = 8 clocks early, 0 = 16 clocks early.
6	0x1	DIS_STP_OB_CLK_DURING_NON_WR: [PMC] 1 = Disables stopping outbound data clock to I/O during non-write transactions.

Bit	Reset	Description
2	0x1	REF_AFTER_SREF: [PMC] 1 = Enables trigger of refresh after exiting self-refresh.
1	ENABLED	CLKCHANGE_PD_ENABLE: [PMC] Forces DRAM into power-down during CLKCHANGE. 0 = DISABLED 1 = ENABLED
0	ENABLED	CLKCHANGE_REQ_ENABLE: [PMC] Allows the EMC and CAR to handshake on PLL divider/source changes. 0 = DISABLED 1 = ENABLED

20.11.2.78 EMC_CFG_DIG_DLL_0

Configure Digital DLL

Controls for digital DLL's, which used to measure and maintain 1/4 cycle phase adjustment for RDQS strobes.

This register is shadowed: see usage note at top.

Boot requirements

- This register (except for field CFG_DLL_ALARM_DISABLE and DLL_RESET (trigger) and CFG_DLL_USE_OVERRIDE_UNTIL_LOCK(trigger)) should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register (except for field CFG_DLL_ALARM_DISABLE and DLL_RESET (trigger) and CFG_DLL_USE_OVERRIDE_UNTIL_LOCK(trigger)) should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2bc | Read/Write: R/W | Reset: 0xXX0000Xd (0bx0000x00000000000000000010x11x1

Bit	R/W	Reset	Description
31	RO	X	CFG_DLL_USE_OVERRIDE_UNTIL_LOCK: [PMC] Writing 1 to this register causes override_val to be used in place of DLL output until DLL_LOCK/DLL_LOCK_TIMEOUT is obtained. Takes effect on next shadow update.
30	RO	0x0	DLL_RESET: [PMC] Writing 1 to this register will send reset pulse to DLL's on next shadow update. Must reset DLLs when changing clock frequency by factor >= 2
29:28	RW	0x0	CFG_DLL_LOCK_LIMIT: [PMC] Reserved in case DLL has problems locking. DLL will be treated as locked after LIMIT µs. Counter is reset with DLL_RESET (from above) or with each periodic update (if using RUN_PERIODIC). Settings are: 00: LIMIT = 16 µs 01: LIMIT = 64 µs 10: LIMIT = 128 µs 11: LIMIT = 512 µs
27	RW	0x0	CFG_DLL_ALARM_DISABLE: [PMC] Reserved bit -- disable override of DLL logic when DLL_ALM is set (otherwise overrides DLI to 0x3FF).
26	RO	X	CFG_DLL_UPDATE_AT_NXT_REFRESH: Writing 1 to this register causes the DLCELL update to occur at the next REFRESH interval. Ordinarily, updates are only performed if the DLL value is updated. This mechanism allows updates to the XFORM_MULT/OFF to be taken at the next REFRESH.
25:16	RW	0x0	CFG_DLL_OVERRIDE_VAL: [PMC] Value to use in place of DLI output if CFG_DLL_OVERRIDE_EN is set or prior to lock in RUN_TIL_LOCK mode.
15	RW	0x0	CFG_DLL_TESTEN: [PMC] Enable DLL test mode
14	RW	0x0	CFG_DLL_QUSE_TESTOUT: [PMC] Enable DLL TESTOUT on QUSE pads
13:12	RW	0x0	CFG_DLL_TESTSEL: [PMC] Select DLL test output
11:8	RW	0x0	CFG_DLL_UDSET: [PMC] DLL Loop filter control (2^(udset+3)).

Bit	R/W	Reset	Description
7:6	RW	RUN_TIL_LOCK	CFG_DLL_MODE: [PMC] Controls how frequently DLL runs, as follows 0 = RUN_CONTINUOUS: DLL will run continuously. This option will consume the most power. Once LOCK/TIMEOUT occurs, trimmers will be updated at REFRESH or shadow update. 1 = RUN_TIL_LOCK: after DLL_RESET is set, DLL will run until it has locked/timed out, then be disabled. Trimmers updated at following REFRESH/shadow update. 2 = RUN_PERIODIC: after DLL_LOCK/TIMEOUT, DLL will be disabled and re-enabled after CFG_DLL_RUN_PERIOD µs to track delay changes (temperature/voltage). 3 = RESERVED
5	RW	0x0	CFG_DLL_LOWSPEED: [PMC] Enable DLL for use with low-speed EMCCLK operation (< 200 MHz).
4	RO	X	CFG_DLL_STALL_RW_UNTIL_LOCK: [PMC] Writing 1 to this register will cause the EMC to stall all RW traffic until the DLL locks. Takes effect on the next shadow update.
3	RW	0x1	CFG_DLL_STALL_ALL_TRAFFIC: [PMC] Enables stalling of all DRAM traffic while undergoing dll_stall condition;otherwise only block reads/writes are stalled during dll_stall
2	RW	ENABLED	CFG_DLL_OVERRIDE_EN: [PMC] Override DLL's DLI output with OVERRIDE_VAL (still uses mult/offset). 0 = DISABLED 1 = ENABLED
0	RW	ENABLED	CFG_DLL_EN: [PMC] Enables digital DLL 0 = DISABLED 1 = ENABLED

20.11.2.79 EMC_CFG_DIG_DLL_PERIOD_0

This register is shadowed: see usage note at top.

Offset: 0x2c0 | Read/Write: R/W | Reset: 0x00008000 (0bxxxxxxxxxxxxxxxx1000000000000000)

Bit	Reset	Description
15:0	0x8000	CFG_DLL_RUN_PERIOD: [PMC] If CFG_DLL_MODE == RUN_PERIODIC, this specifies the interval between runs in microseconds.

20.11.2.80 EMC_DIG_DLL_STATUS_0

Digital DLL Status

Digital DLL Status bits directly from the DLL

Offset: 0x2c8 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	DLL_LOCK
14	X	DLL_ALARM
13	X	DLL_LOCK_TIMEOUT
9:0	X	DLL_OUT

20.11.2.81 EMC_RDV_MASK_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2cc | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	RDV_MASK: [PMC] This should be programmed to RDV.

20.11.2.82 EMC_WDV_MASK_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2d0 | Read/Write: R/W | Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3:0	0xf	WDV_MASK: [PMC] This should be programmed to WDV.

20.11.2.83 EMC_CTT_DURATION_0

DRAM timing parameter

CTT_DURATION controls how long CTT remains enabled. If CTT + CTT_DURATION - RDV - 2 > 0, R2W must be increased by that amount.

Offset: 0x2d8 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx011)

Bit	Reset	Description
2:0	0x3	CTT_DURATION: [PMC] Determines how long CTT remains enabled during reads. CTT determines when it will be enabled. DDR3: CTT_TERMINATION == 1 ? 4 : 0 LPDDR2: 0

20.11.2.84 EMC_CTT_TERM_CTRL_0

Configure CTT Termination Output Drive Strength

If CTT_TERMINATION is enabled, this controls the strength of the output drivers.

TERM_OVERRIDE forces use of EMC2TMC_CFG*_DRVDN_TERM, EMC2TMC_CFG*_DRVUP_TERM instead of using of AUTO_CAL DRVDN/DRVUP values with the following equation applied:

```
TERM_DRDN = (AUTO_CAL_DRVDN * TERM_SLOPE) / 4 + TERM_OFFSET
TERM_DRUP = (AUTO_CAL_DRVUP * TERM_SLOPE) / 4 + TERM_OFFSET
```

Recommended values for SLOPE/OFFSET:

100ohm CTT: TERM_SLOPE = 0x1, TERM_OFFSET = 0x4

50ohm CTT: TERM_SLOPE = 0x2, TERM_OFFSET = 0x8

Note: setting slope = 0 allows TERM_DRDN/DRVUP = TERM_OFFSET

setting slope > 2 is not supported

Boot requirements

- This register (except for fields TERM_DRVDN and TERM_DRVUP) should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register (except for fields TERM_DRVDN and TERM_DRVUP) should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2dc | Read/Write: R/W | Reset: 0xXX0XX802 (0b0xxxxxxxxxxxxxxxxxx01000xxxxx010)

Bit	R/W	Reset	Description
31	RW	DISABLED	TERM_OVERRIDE: [PMC] 0 = DISABLED 1 = ENABLED
28:24	RO	X	TERM_DRVUP
19:15	RO	X	TERM_DRVDN
12:8	RW	0x8	TERM_OFFSET: [PMC]
2:0	RW	0x2	TERM_SLOPE: [PMC]

ZQ Calibration Registers -- LPDDR2/DDR3

1. Periodic mode

Allows a ZQ calibration command to be sent periodically to DRAM. After ZCAL_INTERVAL, ZCAL_MRW_CMD will be sent to each DRAM, 1 at a time, followed by ZCAL_WAIT_CNT interval in which no other commands will be allowed to be sent to either DRAM. So if ZQ_MRW_DEV_SELECTN == 2'b00, it would send the MRW command to dev0, wait ZCAL_WAIT_CNT, send the command to dev1, wait ZCAL_WAIT_CNT, resume normal operation. It will then wait ZCAL_INTERVAL microseconds before repeating the procedure. The ZQ calibration command will not be sent to devices that are 1) unpopulated and 2) either in or about to enter self-refresh or DPD.

To enable periodic ZQ calibration, program ZCAL_INTERVAL and ZCAL_WAIT_CNT to be non-zero, as well as ZCAL_MRW_CMD to contain the command and device selection, followed by TIMING_CONTROL to latch those three registers.

Note that ZCAL_WAIT_CNT will be used instead of MRS_WAIT_CNT for delaying the subsequent DRAM commands after the ZQ calibration commands, even though in LPDDR2 a ZQ calibration command is also an MRW command.

Disable this feature by setting ZCAL_INTERVAL.ZCAL_REF_INTERVAL = 0, followed by TIMING_CONTROL to latch it.

2. One-shot mode

Allows ZQ calibration command to be sent to DRAM.

To send a one-shot command, keep ZCAL_INTERVAL and ZCAL_WAIT_CNT equal to zero, program ZCAL_MRW_CMD with only one device selected, then program TIMING_CONTROL to latch in those registers, and finally program ZCAL_ONE_SHOT to 1 to trigger the one-shot command. If more than one device needs to be calibrated, wait the ZQ calibration time and repeat the steps above with the other device selection bit enabled in ZCAL_MRW_CMD.

Do not send a one-shot ZQ calibration command when periodic mode is enabled.

20.11.2.85 EMC_ZCAL_INTERVAL_0

Configure ZQ Calibration

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:10	0x0	ZCAL_INTERVAL_HI: [PMC] Combined with ZCAL_INTERVAL_LO specifies the number of microseconds to wait between issuance of ZCAL_MRW_CMD. If 0, ZCAL is disabled and internal counter will be reset.
9:0	0x0	ZCAL_INTERVAL_LO [PMC]

20.11.2.86 EMC_ZCAL_WAIT_CNT_0

Configure ZQ Calibration

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	ZCAL_WAIT_CNT: [PMC] Number of EMC clocks to wait before issuing any commands after sending ZCAL_MRW_CMD.

20.11.2.87 EMC_ZCAL_MRW_CMD_0

Configure ZQ Calibration

This register is shadowed: see usage note at top.

(However it does not obey READ_MUX/WRITE_MUX non-default values.)

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2e8 | Read/Write: WO | Reset: 0x00000000 (0b00xxxxxx00000000xxxxxxxx00000000)

Bit	Reset	Description
31:30	0x0	ZQ_MRW_DEV_SELECTN: [PMC] Active-low chip-select, 0x0 applies command to both devices (will happen 1 at a time), 0x2 to for only dev0, 0x1 for dev1.
23:16	0x0	ZQ_MRW_MA: [PMC] MRW MA field to be sent after ZCAL_INTERVAL

Bit	Reset	Description
7:0	0x0	ZQ_MRW_OP: [PMC] MRW OP field to be sent after ZCAL_INTERVAL

20.11.2.88 EMC_ZQ_CAL_0

Trigger a Single ZQ Calibration

This register issues a ZQ calibration command for DDR3.

Offset: 0x2ec | Read/Write: R/W | Reset: 0x00000010 (0b00xxxxxxxxxxxxxxxxxxxxxxxxx1xxx0)

Bit	Reset	Description
31:30	0x0	ZQ_CAL_DEV_SELECTN: Active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1, 0x3 for neither device (0x0, for both devices, is not allowed if the ZQ resister is shared between devices).
4	LONG	ZQ_CAL_LENGTH: Indicate short or long ZQ calibration. 0 = SHORT 1 = LONG
0	0x0	ZQ_CAL_CMD: Issues a ZQ calibration command (DDR3 only).

20.11.2.89 EMC_XM2CMDPADCTRL_0

Pad configuration registers from APB_MISC:

XM2CMD Pad Control Register

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2f0 | Read/Write: R/W | Reset: 0x00111200 (0bxxxxxxxxxx001x001x001x01x00000xxx)

Bit	Reset	Description
22:20	0x1	EMC2TMC_CFG_XM2CMD_CLKBUF2_CONFIG: [PMC] CONFIG bit setting of CLKBUF2 pad (000 == DLCELL mode, 001 == TRIM/PDL mode
18:16	0x1	EMC2TMC_CFG_XM2CMD_CLKBUF1_CONFIG: [PMC] CONFIG bit setting of the CLKBUF1 pad (000 == DLCELL mode, 001 == TRIM/PDL mode)
14:12	0x1	EMC2TMC_CFG_XM2CMD_CLKBUF0_CONFIG: [PMC] CONFIG bit setting of the CLKBUF0 pad (000 == DLCELL mode, 001 == TRIM/PDL mode)
10	NORMAL	CFG_XM2CMD_CAL_SELECT: [PMC] Choose VREF/NORMAL (DQ/DQS) calibration for CMD pads. 0 = NORMAL 1 = VREF
9	ENABLE	EMC2TMC_CFG_XM2RESET_E_DDR3: [PMC] Select pad mode 0 = DISABLE 1 = ENABLE
7	DISABLE	EMC2TMC_CFG_XM2CMD_E_DDR3: [PMC] Select pad mode 0 = DISABLE 1 = ENABLE
6	0x0	EMC2TMC_CFG_XM2CMD_CLK_SEL: [PMC] pad clk_sel (ma bits get this value inverted in LPDDR2 mode)
5	0x0	EMC2TMC_CFG_XM2CMD_E_PREEMP: [PMC] XM2CMD data pins pre-emp enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x0	EMC2TMC_CFG_XM2CMD_E_BYPASS: [PMC] XM2CMD pins bypass outbound flop enable 0 = DISABLE 1 = ENABLE
3	0x0	EMC2TMC_CFG_XM2CMD_E_PWRD: [PMC] XM2CMD pins pad power-down signal 0 = DISABLE 1 = ENABLE

20.11.2.90 EMC_XM2CMDPADCTRL2_0

Offset: 0x2f4 | Read/Write: R/W | Reset: 0x050c0000 (0b00000101000011000xxxxxxxxxxxxx) | Default: 0x858c0000

Bit	Reset	SW Default	Description
31:28	0x0	0x8	EMC2PMACRO_CFG_XM2CMD_DRVUP_SLWF [PMC]
27:24	0x5	0x5	EMC2PMACRO_CFG_XM2CMD_DRVDN_SLWR [PMC]
23:19	0x1	0x11	EMC2PMACRO_CFG_XM2CMD_DRVUP [PMC]
18:14	0x10	0x10	EMC2PMACRO_CFG_XM2CMD_DRVDN [PMC]

20.11.2.91 EMC_XM2DQSPADCTRL_0

XM2DQS Pad Control Register

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2f8 | Read/Write: R/W | Reset: 0x490c1414 (0b01001001000011000x10100xxx10100) | Default: 0x888c1414

Bit	Reset	SW Default	Description
31:28	0x4	0x8	EMC2PMACRO_CFG_XM2DQS_DRVUP_SLWF [PMC]
27:24	0x9	0x8	EMC2PMACRO_CFG_XM2DQS_DRVDN_SLWR [PMC]
23:19	0x1	0x11	EMC2PMACRO_CFG_XM2DQS_DRVUP [PMC]
18:14	0x10	0x10	EMC2PMACRO_CFG_XM2DQS_DRVDN [PMC]
12:8	0x14	0x14	EMC2PMACRO_CFG_XM2DQS_DRVUP_TERM [PMC]
4:0	0x14	0x14	EMC2PMACRO_CFG_XM2DQS_DRVDN_TERM [PMC]

20.11.2.92 EMC_XM2DQSPADCTRL2_0

XM2DQS Pad Control Register

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x2fc | Read/Write: R/W | Reset: 0x0000be18 (0bxxxxxxxx0xx00000010111110000011000)

Bit	Reset	Description
24	0x0	EMC2TMC_CFG_XM2DQS_QUSE_DLL_BYP: [PMC] 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
21:20	0x0	EMC2TMC_CFG_XM2DQS_CONFIG: [PMC] IOBRICK rfu pins
19:18	0x0	EMC2TMC_CFG_XM2DQS_RX_DQS_SEL: [PMC] IOBRICK rfu pins
17:16	0x0	EMC2TMC_CFG_XM2DQS_RX_DQ_SEL: [PMC] [0] = select between legacy mode (0) or LSSA mode (1) DQ RX. [1]=RFU
15	0x1	EMC2TMC_CFG_XM2DQS_E_SCHMT_DQ: [PMC] XM2DQ DQ pins Schmitt enable 0 = DISABLE 1 = ENABLE
14	0x0	EMC2TMC_CFG_XM2DQS_E_PWRD: [PMC] XM2DQS pins pad power-down signal 0 = DISABLE 1 = ENABLE
13	0x1	EMC2TMC_CFG_XM2DQS_E_SCHMT_DQS: [PMC] XM2DQS DQS pins Schmitt enable 0 = DISABLE 1 = ENABLE
12	DISABLE	EMC2TMC_CFG_XM2DQS_QUSE_DLI_EN: [PMC] Active low 0 = ENABLE 1 = DISABLE
11	DISABLE	EMC2TMC_CFG_XM2DQS_TXDQS_DLI_EN: [PMC] Active low 0 = ENABLE 1 = DISABLE
10	DISABLE	EMC2TMC_CFG_XM2DQS_TXDQ_DLI_EN: [PMC] Active low 0 = ENABLE 1 = DISABLE
9	DISABLE	EMC2TMC_CFG_XM2DQS_RX_DLI_EN: [PMC] Active low 0 = ENABLE 1 = DISABLE
8	DISABLE	EMC2TMC_CFG_XM2DQS_E_DDR3: [PMC] Select pad mode 0 = DISABLE 1 = ENABLE
7	0x0	EMC2TMC_CFG_XM2DQS_CLKSEL_DQS: [PMC]
6	0x0	EMC2TMC_CFG_XM2DQS_CLKSEL_DQ: [PMC]
5	0x0	EMC2TMC_CFG_XM2DQS_E_VREF_DQ: [PMC] 0 = DISABLE 1 = ENABLE
4	ENABLE	EMC2TMC_CFG_XM2DQS_E_CTT_HIZ_DQS: [PMC] 0 = DISABLE 1 = ENABLE
3	ENABLE	EMC2TMC_CFG_XM2DQS_E_CTT_HIZ_DQ: [PMC] 0 = DISABLE 1 = ENABLE
2	0x0	EMC2TMC_CFG_XM2DQS_E_PREEMP: [PMC] CFG_XM2DQ data pins pre-emp enable 0 = DISABLE 1 = ENABLE
1	0x0	EMC2TMC_CFG_XM2DQS_E_BYPASS: [PMC] CFG_XM2DQ data pins bypass outbound flop enable 0 = DISABLE 1 = ENABLE
0	0x0	EMC2TMC_CFG_XM2DQS_E_RX_FT_REC: [PMC] 0 = DISABLE 1 = ENABLE

20.11.2.93 EMC_XM2DQPADCTRL_0

CFG_XM2DQ Pad Control Register

These values only used if autocal is disabled

Offset: 0x300 | Read/Write: R/W | Reset: 0x490c2990 (0b01001001000011000001010011001xxxx) | Default: 0x888c2990

Bit	Reset	SW Default	Description
31:28	0x4	0x8	EMC2PMACRO_CFG_XM2DQ_DRVUP_SLWF [PMC]
27:24	0x9	0x8	EMC2PMACRO_CFG_XM2DQ_DRVDN_SLWR [PMC]
23:19	0x1	0x11	EMC2PMACRO_CFG_XM2DQ_DRVUP [PMC]
18:14	0x10	0x10	EMC2PMACRO_CFG_XM2DQ_DRVDN [PMC]
13:9	0x14	0x14	EMC2PMACRO_CFG_XM2DQ_DRVUP_TERM [PMC]
8:4	0x19	0x19	EMC2PMACRO_CFG_XM2DQ_DRVDN_TERM [PMC]

20.11.2.94 EMC_XM2DQPADCTRL2_0

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x304 | Read/Write: R/W | Reset: 0x00000000 (0bx000x000x000x000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
30:28	0x0	EMC2TMC_CFG_XM2DQ3_DLYIN_TRM: [PMC3] delay trim for byte 3
26:24	0x0	EMC2TMC_CFG_XM2DQ2_DLYIN_TRM: [PMC3] delay trim for byte 2
22:20	0x0	EMC2TMC_CFG_XM2DQ1_DLYIN_TRM: [PMC3] delay trim for byte 1
18:16	0x0	EMC2TMC_CFG_XM2DQ0_DLYIN_TRM: [PMC3] delay trim for byte 0

20.11.2.95 EMC_XM2CLKPADCTRL_0

XM2CLK Pad Control Register

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x308 | Read/Write: R/W | Reset: 0xffffc000 (0b11111111111111111110000000xx000x0)

Bit	Reset	Description
31:28	0xf	EMC2PMACRO_CFG_XM2CLK_DRVUP_SLWF [PMC]
27:24	0xf	EMC2PMACRO_CFG_XM2CLK_DRVDN_SLWR [PMC]
23:19	0x1f	EMC2PMACRO_CFG_XM2CLK_DRVUP: [PMC] also used for the override of the second auto-calibration cycle
18:14	0x1f	EMC2PMACRO_CFG_XM2CLK_DRVDN: [PMC] also used for the override of the second auto-calibration cycle
7	DISABLE	EMC2TMC_CFG_XM2CLK_E_DDR3: [PMC] Select pad mode 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	DISABLE	EMC2TMC_CFG_XM2CLK_E_PWRD: [PMC] XM2CLK pins pad power-down signal 0 = DISABLE 1 = ENABLE
3	DISABLE	EMC2TMC_CFG_XM2CLK_E_CAL_BYPASS: [PMC] XM2CLK bypass drvdown/up calibration 0 = DISABLE 1 = ENABLE
2	DISABLE	EMC2TMC_CFG_XM2CLK_E_PREEMP: [PMC] pre-emphasis enable 0 = DISABLE 1 = ENABLE
0	DISABLE	EMC2PMACRO_CFG_DYN_PULLS_ON_CLKDIS: [PMC] Enable pull-up on CLKP and pull-down on CLKN when disabling clock. Reserved in case clock startup ramp is too slow (DDR3). 0 = DISABLE 1 = ENABLE

20.11.2.96 EMC_XM2COMPPADCTRL_0

MEM_COMP Pad Control Register

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x30c | Read/Write: R/W | Reset: 0x81f1f008 (0b1000xxx11111xxx111110000000x1000

Bit	Reset	Description
24:20	0x1f	CFG_XM2COMP_DRVUP: [PMC] used if AUTOCAL is disabled
16:12	0x1f	CFG_XM2COMP_DRVDN: [PMC] used if AUTOCAL is disabled
11	DISABLE	EMC2TMC_CFG_XM2COMP_E_TESTOUT: [PMC] 0 = DISABLE 1 = ENABLE
10	DISABLE	CFG_XM2COMP_VREF_CAL_EN: [PMC] When enabled (=1), generate MCLK calibration cycle. 0 = DISABLE 1 = ENABLE
9	DISABLE	EMC2TMC_CFG_XM2COMP_E_PWRD: [PMC] XM2CLK pins pad power-down signal 0 = DISABLE 1 = ENABLE
8	DISABLE	EMC2TMC_CFG_XM2COMP_E_DDR3: [PMC] Select pad mode 0 = DISABLE 1 = ENABLE
7:5	0x0	EMC2TMC_CFG_XM2COMP_BIAS_SEL: [PMC]
3:0	0x8	EMC2TMC_CFG_XM2COMP_PD_VREF_SEL: [PMC]

20.11.2.97 EMC_XM2VTTGENPADCTRL_0

XM2 MISC/VTTGEN Pad Control Register

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x310 | Read/Write: R/W | Reset: 0x00005500 (0bxxxxx000xxxxx000x101x101xxxxx0xx)

Bit	Reset	Description
26:24	0x0	EMC2TMC_CFG_XM2VTTGEN_DRVUP: [PMC]
18:16	0x0	EMC2TMC_CFG_XM2VTTGEN_DRVDN: [PMC]
14:12	0x5	EMC2TMC_CFG_XM2VTTGEN_VAUXP_LEVEL: [PMC]
10:8	0x5	EMC2TMC_CFG_XM2VTTGEN_VCLAMP_LEVEL: [PMC]
2	0x0	EMC2TMC_CFG_XM2VTTGEN_E_DDR3: [PMC] Select pad mode

20.11.2.98 EMC_XM2VTTGENPADCTRL2_0

- E_NO_VTTGEN: disables VTTGEN altogether. One VTTGEN pad in each MEM section is always enabled (DDR3-only)

Offset: 0x314 | Read/Write: R/W | Reset: 0x0000003f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx111111)

Bit	Reset	Description
5:0	0x3f	EMC2TMC_CFG_XM2VTTGEN_E_NO_VTTGEN: [PMC] Disable optional VTTGEN pads (1 bit per pad) [0]--ADDR0, [1]--ADDR1, [2]--ADDR2, [3]--DATA1, [4]--DATA2, [5]--DATA3

20.11.2.99 EMC_EMCPADEN_0

EMC pad enable register

Writing 0 will disable listed direction

Offset: 0x31c | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11)

Bit	Reset	Description
1	0x1	EMC2PMACRO_CFG_PAD_INPUT_EN: Inputs enable for EMC pads 0 = DISABLE 1 = ENABLE
0	0x1	EMC2PMACRO_CFG_PAD_OUTPUT_EN: Outputs enable for EMC pads 0 = DISABLE 1 = ENABLE

20.11.2.100 EMC_XM2DQSPADCTRL4_0

Offset: 0x320 | Read/Write: R/W | Reset: 0x00208208 (0bxxxxxxxxx01000x01000x01000x01000)

Bit	Reset	Description
22:18	0x8	EMC2TMC_CFG_XM2DQS_BYTE3_VREF_DQ: [PMC]
16:12	0x8	EMC2TMC_CFG_XM2DQS_BYTE2_VREF_DQ: [PMC]
10:6	0x8	EMC2TMC_CFG_XM2DQS_BYTE1_VREF_DQ: [PMC]
4:0	0x8	EMC2TMC_CFG_XM2DQS_BYTE0_VREF_DQ: [PMC]

20.11.2.101 EMC_SCRATCH0_0

This is a scratch register for general use.

Offset: 0x324 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH:

20.11.2.102 EMC_DLL_XFORM_DQS0_0

Configure Digital DLL

XFORM_DQSx_MULT and XFORM_DQSx_OFFS are a multiplier and offset, respectively, that can be used to modify the DLL output. Final products may be read out in DQS_TRIMMER_RD register. The default is multiply by 1 to keep the DLL's 1/4 cycle delay. Integer + fractional formats for multiplier and offset:

OFFS is 2's complement format, with bit [0] representing integer

The output of the XFORM is xform_out = (xform_in * MULT + OFFS*16)/ 16

Largest allowed MULT = 0x17

In the case of overflow, output is clamped according to these rules:

out >= 0x600, xform_out = 0x000

0x600 > out >= 0x400, xform_out = 0x3ff

That means clamping works within 50% above the maximum (0x3ff) DLL output or 50% below the minimum (0) DLL output.

To make sure that is always satisfied, conservatively program OFFS so that 0x600 < OFFS < (0x5ff – ((MULT * 0x3ff) >> 4)). The DLL output is effectively overridden by setting XFORM_*_MULT to 0 and programming XFORM_*_OFFS[8:1] to override value.

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

The DQS_CURRENT_TRIM_VAL_BYTE_x registers provide a way to observe the values being used by the trimmers

Offset: 0x328 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_DQS0_OFFS: [PMC]
4:0	0x10	XFORM_DQS0_MULT: [PMC]

20.11.2.103 EMC_DLL_XFORM_DQS1_0

Configure Digital DLL

Offset: 0x32c | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_DQS1_OFFS: [PMC]
4:0	0x10	XFORM_DQS1_MULT: [PMC]

20.11.2.104 EMC_DLL_XFORM_DQS2_0

Configure Digital DLL

Offset: 0x330 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_DQS2_OFFS: [PMC]
4:0	0x10	XFORM_DQS2_MULT: [PMC]

20.11.2.105 EMC_DLL_XFORM_DQS3_0

Configure Digital DLL

Offset: 0x334 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_DQS3_OFFS: [PMC]
4:0	0x10	XFORM_DQS3_MULT: [PMC]

20.11.2.106 EMC_DLL_XFORM_DQS4_0

Configure Digital DLL

Offset: 0x338 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_DQS4_OFFS: [PMC]
4:0	0x10	XFORM_DQS4_MULT: [PMC]

20.11.2.107 EMC_DLL_XFORM_DQS5_0

Configure Digital DLL

Offset: 0x33c | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_DQS5_OFFS: [PMC]
4:0	0x10	XFORM_DQS5_MULT: [PMC]

20.11.2.108 EMC_DLL_XFORM_DQS6_0

Configure Digital DLL

Offset: 0x340 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_DQS6_OFFS: [PMC]
4:0	0x10	XFORM_DQS6_MULT: [PMC]

20.11.2.109 EMC_DLL_XFORM_DQS7_0

Configure Digital DLL

Offset: 0x344 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_DQS7_OFFS: [PMC]
4:0	0x10	XFORM_DQS7_MULT: [PMC]

20.11.2.110 EMC_DLL_XFORM_QUSE0_0

Configure Digital DLL

XFORM_QUSEx_MULT and XFORM_QUSEx_OFFS are a multiplier and offset, respectively, that can be used to modify the DLL output. Final product may be read out in DQS_TRIMMER_RD register. The default is multiply by 1/2, to give 1/8 cycle delay.

OFFS is 2's complement format, with bit [0] representing integer. The output of the XFORM is xform_out = (xform_in * MULT + OFFS*16)/ 16

Largest allowed MULT = 0x17

In the case of overflow, output is clamped according to these rules:

out >= 0x600, xform_out = 0x000

0x600 > out >= 0x400, xform_out = 0x3ff

That means clamping works within 50% above the maximum (0x3ff) DLL output or 50% below the minimum (0) DLL output. To make sure that is always satisfied, conservatively program OFFS so that 0x600 < OFFS < (0x5ff – ((MULT * 0x3ff) >> 4)). The DLL output is effectively overridden by setting XFORM_*_MULT to 0 and programming XFORM_*_OFFS[8:1] to override value.

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x348 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxx00000000000xxxxxxx01000)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE0_OFFS: [PMC]
4:0	0x8	XFORM_QUSE0_MULT: [PMC]

20.11.2.111 EMC_DLL_XFORM_QUSE1_0

Configure Digital DLL

Offset: 0x34c | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxx00000000000xxxxxxx01000)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE1_OFFS: [PMC]
4:0	0x8	XFORM_QUSE1_MULT: [PMC]

20.11.2.112 EMC_DLL_XFORM_QUSE2_0

Configure Digital DLL

Offset: 0x350 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxx00000000000xxxxxxx01000)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE2_OFFS: [PMC]
4:0	0x8	XFORM_QUSE2_MULT: [PMC]

20.11.2.113 EMC_DLL_XFORM_QUSE3_0

Configure Digital DLL

Offset: 0x354 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxx00000000000xxxxxxx01000)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE3_OFFS: [PMC]
4:0	0x8	XFORM_QUSE3_MULT: [PMC]

20.11.2.114 EMC_DLL_XFORM_QUSE4_0

Configure Digital DLL

Offset: 0x358 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxx00000000000xxxxxxx01000)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE4_OFFS: [PMC]
4:0	0x8	XFORM_QUSE4_MULT: [PMC]

20.11.2.115 EMC_DLL_XFORM_QUSE5_0

Configure Digital DLL

Offset: 0x35c | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxx00000000000xxxxxxx01000)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE5_OFFS: [PMC]
4:0	0x8	XFORM_QUSE5_MULT: [PMC]

20.11.2.116 EMC_DLL_XFORM_QUSE6_0

Configure Digital DLL

Offset: 0x360 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxx00000000000xxxxxxx01000)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE6_OFFS: [PMC]
4:0	0x8	XFORM_QUSE6_MULT: [PMC]

20.11.2.117 EMC_DLL_XFORM_QUSE7_0

Configure Digital DLL

Offset: 0x364 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxx00000000000xxxxxxx01000)

Bit	Reset	Description
22:12	0x0	XFORM_QUSE7_OFFS: [PMC]
4:0	0x8	XFORM_QUSE7_MULT: [PMC]

20.11.2.118 EMC_DLL_XFORM_DQ0_0

Configure Digital DLL

XFORM_DQx_MULT and XFORM_DQx_OFFS are a multiplier and offset, respectively, that can be used to modify the DLL output. XFORM_DQx_* applies to all DQ/DM bits in each byte x. Final products may be read out in the DQ_TRIMMER_RD register. The default is multiply by 1, providing 1/4 cycle skew to DQS to keep DLL's 1/4 cycle delay. Integer + fractional formats for multiplier & offset:

OFFS is 2's complement format, with bit [0] representing integer. The output of the XFORM is xform_out = (xform_in * MULT + OFFS*16)/ 16

Largest allowed MULT = 0x17

In the case of overflow, output is clamped according to these rules:

out >= 0x600, xform_out = 0x000

0x600 > out >= 0x400, xform_out = 0x3ff

That means clamping works within 50% above the maximum (0x3ff) DLL output or 50% below the minimum (0) DLL output. To make sure that is always satisfied, conservatively program OFFS so that 0x600 < OFFS < (0x5ff – ((MULT * 0x3ff) >> 4)).

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x368 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_TXDQ0_OFFS: [PMC]
4:0	0x10	XFORM_TXDQ0_MULT: [PMC]

20.11.2.119 EMC_DLL_XFORM_DQ1_0

Configure Digital DLL

Offset: 0x36c | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_TXDQ1_OFFS: [PMC]
4:0	0x10	XFORM_TXDQ1_MULT: [PMC]

20.11.2.120 EMC_DLL_XFORM_DQ2_0

Configure Digital DLL

Offset: 0x370 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_TXDQ2_OFFS: [PMC]
4:0	0x10	XFORM_TXDQ2_MULT: [PMC]

20.11.2.121 EMC_DLL_XFORM_DQ3_0

Configure Digital DLL

Offset: 0x374 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxx00000000000xxxxxxx10000)

Bit	Reset	Description
22:12	0x0	XFORM_TXDQ3_OFFS: [PMC]
4:0	0x10	XFORM_TXDQ3_MULT: [PMC]

20.11.2.122 EMC_DLI_RX_TRIM0_0

Configure Digital DLL

Offset: 0x378 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_0
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_0

20.11.2.123 EMC_DLI_RX_TRIM1_0

Configure Digital DLL

Offset: 0x37c | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_1
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_1

20.11.2.124 EMC_DLI_RX_TRIM2_0

Configure Digital DLL

Offset: 0x380 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_2
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_2

20.11.2.125 EMC_DLI_RX_TRIM3_0

Configure Digital DLL

Offset: 0x384 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_3
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_3

20.11.2.126 EMC_DLI_RX_TRIM4_0

Configure Digital DLL

Offset: 0x388 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_4
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_4

20.11.2.127 EMC_DLI_RX_TRIM5_0

Configure Digital DLL

Offset: 0x38c | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_5
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_5

20.11.2.128 EMC_DLI_RX_TRIM6_0

Configure Digital DLL

Offset: 0x390 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_6
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_6

20.11.2.129 EMC_DLI_RX_TRIM7_0

Configure Digital DLL

Offset: 0x394 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_7
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_7

20.11.2.130 EMC_DLI_TX_TRIM0_0

Configure Digital DLL

Offset: 0x398 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_0

20.11.2.131 EMC_DLI_TX_TRIM1_0

Configure Digital DLL

Offset: 0x39c | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_1

20.11.2.132 EMC_DLI_TX_TRIM2_0

Configure Digital DLL

Offset: 0x3a0 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_2

20.11.2.133 EMC_DLI_TX_TRIM3_0

Configure Digital DLL

Offset: 0x3a4 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_3

20.11.2.134 EMC_DLI_TRIM_TXDQS0_0

Set DLI TRIM

DLI_TRIM_TXDQSx

Controls trimmer used to skew data byte DQS/DQ output relative to MCK. 0-3 apply to rank 0, 4-7 apply to rank 1 (if dual-rank enabled)

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x3a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS0_DLI: [PMC]

20.11.2.135 EMC_DLI_TRIM_TXDQS1_0

Set DLI TRIM

Offset: 0x3ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS1_DLI: [PMC]

20.11.2.136 EMC_DLI_TRIM_TXDQS2_0

Set DLI TRIM

Offset: 0x3b0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS2_DLI: [PMC]

20.11.2.137 EMC_DLI_TRIM_TXDQS3_0

Set DLI TRIM

Offset: 0x3b4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS3_DLI: [PMC]

20.11.2.138 EMC_DLI_TRIM_TXDQS4_0

Set DLI TRIM

Offset: 0x3b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS4_DLI: [PMC]

20.11.2.139 EMC_DLI_TRIM_TXDQS5_0

Set DLI TRIM

Offset: 0x3bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS5_DLI: [PMC]

20.11.2.140 EMC_DLI_TRIM_TXDQS6_0

Set DLI TRIM

Offset: 0x3c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS6_DLI: [PMC]

20.11.2.141 EMC_DLI_TRIM_TXDQS7_0

Set DLI TRIM

Offset: 0x3c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS7_DLI: [PMC]

Scheme to change controller timing parameters + SDRAM mode timing atomically.

1. Program the EMC controller timing registers (shadowed).
2. Pre-program the SDRAM mode registers.
    a. Write to whatever EMC registers you want to be execute after clock change request has been detected but before the actual clock change happens. For example, if you want to disable DLL in DDR3 mode, you can,
        i. Issue a write to SDRAM's MR1 register via EMC MRS register to disable DLL.
        ii. Issue a Self-refresh entry via EMC SELF_REF register.

    Note: If there's no need to execute any register access before clock change, simply skip step (a).

    b. Write 1 to STALL_THEN_EXE_AFTER_CLKCHANGE field which will prevent further config execution until after the actual clock change has happened.
    c. Write to whatever EMC registers you want to be executed after the clock change. Using the same example as (a), you will,
        i. Issue a Self-refresh exit via EMC SELF_REF register.
        ii. Issue writes to SDRAM's MRx register(s) to change read/write latencies and/or enable DLL

iii. Issue burst refreshes via EMC REF register.

Note: If there's no need to execute any register access after clock change, simply skip step (ii-iii); step (b) is still required.

3. Program the CAR to change either clock source and/or clock divider.

Note: In both sequences above, DYN_SELF_REF must be disabled off before the first step, then it can be restored after the last step.

The interval between two clock change sequences are recommended to be at least 20 µs apart

20.11.2.142 EMC_STALL_THEN_EXE_AFTER_CLKCHANGE_0

Write to this register will stall until after EMC timing registers are updated and that clock change has been completed. Once that happens, whatever register access follows this register write will be executed until UNSTALL_RW_AFTER_CLKCHANGE is encountered. At that point, normal memory read/write will be allowed to resume.

Offset: 0x3cc | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	STALL_THEN_EXE_AFTER_CLKCHANGE: Writing a one to this bit will stall subsequent register access.

20.11.2.143 EMC_AUTO_CAL_CLK_STATUS_0

EMC Pad Calibration Status

Offset: 0x3d4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:24	X	AUTO_CAL_CLK_PULLDOWN_ADJ: Pull-down code sent to pads for MCLK pad.
20:16	X	AUTO_CAL_CLK_PULLUP_ADJ: Pull-up code sent to pads for MCLK pad.
12:8	X	AUTO_CAL_CLK_PULLDOWN: Pull-down code generated by auto-calibration for MCLK pad.
4:0	X	AUTO_CAL_CLK_PULLUP: Pull-up code generated by auto-calibration for MCLK pad.

20.11.2.144 EMC_SEL_DPD_CTRL_0

Configures Functional SEL_DPD Modes

Offset: 0x3d8 | Read/Write: R/W | Reset: 0x00040000 (0bxxxxxxxxxxxxxx100xxxxxx00xx0000xx)

Bit	Reset	Description
18:16	0x4	SEL_DPD_DLY: [PMC] number of cycles to wait before asserting SEL_DPD
9	DISABLED	QUSE_SEL_DPD_EN: [PMC] allow SEL_DPD assertion for QUSE pads 0 = DISABLED 1 = ENABLED
8	DISABLED	DATA_SEL_DPD_EN: [PMC] allow SEL_DPD assertion for data pads 0 = DISABLED 1 = ENABLED
5	DISABLED	ODT_SEL_DPD_EN: [PMC] tie SEL_DPD for ODT pads to ENABLE_ODT_DURING_WRITE 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
4	DISABLED	RESET_SEL_DPD_EN: [PMC] assert SEL_DPD for reset pad 0 = DISABLED 1 = ENABLED
3	DISABLED	CA_SEL_DPD_EN: [PMC] allow SEL_DPD assertion for discrete command/address pads 0 = DISABLED 1 = ENABLED
2	DISABLED	CLK_SEL_DPD_EN: [PMC] allow SEL_DPD assertion for discrete clock pad 0 = DISABLED 1 = ENABLED

20.11.2.145 EMC_PRE_REFRESH_REQ_CNT_0

Offset: 0x3dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	PRE_REF_REQ_CNT: [PMC] When the refresh counter reach this pre-refresh request count just before a burst refresh will be issued when it reach the {REFRESH,REFRESH_LO}, a pre-refresh request will be asserted to MC to close the banks/pages and flush its pipeline. A 0 value will disable this feature.

20.11.2.146 EMC_DYN_SELF_REF_CONTROL_0

Threshold for Dynamic Self-Refresh Entry

This register controls how dynamic self-refresh entry/exit is handled.

If ODT is enabled, the DSR_PER_DEVICE field must be set to DISABLED.

It is recommended to change the values of ODT_WRITE and DSR_PER_DEVICE with a clock change

To do it outside of a clock change sequence, one has to be aware that DSR enable/disable takes time to take effect. This is the sequence for disabling DSR_PER_DEVICE and enabling ODT outside of clock change:

1. DYN_SELF_REF <= DISABLED (DSR exit)
2. TIMING_UPDATE <= 1 (latch in the shadow value)
3. Read back and discard any EMC register such as INTSTATUS (ensure 1 and 2 have gone through)
4. Wait 2 µs for the last possible self-refresh entry
5. Poll for SDRAM_IN_SELF_REFRESH == 0 (ensure that self-refresh exits)
6. ODT_WRITE <= new value to turn on ODT

   DSR_PER_DEVICE <= DISABLED (turn off per-device DSR)

   DYN_SELF_REF <= ENABLED (re-enable DSR)
7. TIMING_UPDATE <= 1 (latch in the shadow values)

Offset: 0x3e0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	DISABLED	DSR_PER_DEVICE: [PMC] controls whether dynamic self-refresh is done on a per-device basis. 0 = DISABLED 1 = ENABLED
15:0	0x0	DSR_THRESHOLD: [PMC] number of idle cycles to wait before allowing dynamic self-refresh entry

20.11.2.147 EMC_TXSRDLL_0

DRAM timing parameter

This register is shadowed: see usage note at top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Note: If operating in non-DLL mode, this register needs to be updated with the non-DLL timing requirement.

Offset: 0x3e4 | Read/Write: R/W | Reset: 0x000007ff (0bxxxxxxxxxxxxxxxxxxxxx011111111111)

Bit	Reset	Description
11:0	0x7ff	TXSRDLL: [PMC] Cycles between self-refresh exit and first DRAM command requiring a locked DLL. For DDR3 only: READ (and RAP) and synchronous ODT commands. DDR3: 512. LPDDR2: TXSR (not used). Largest allowed value is 0xffe

20.11.2.148 EMC_CCFIFO_ADDR_0

CCFIFO_ADDR: This register contains the address offset of the EMC register that is intended to be executed during the clock change sequence.

Note: Before triggering a clock change sequence from the CAR, you must configure the pre-/post- clock change sequence by writing, multiple times, to the CCFIFO_DATA/CCFIFO_ADDR register pair. After the CCFIFO_ADDR register is written, Hardware will write the register offset and corresponding data into the 32-deep clock change FIFO (CCFIFO).

After the last CCFIFO_DATA/CCFIFO_ADDR register pair is written for a given clock change sequence, a dummy EMC register read (such as to the CCFIFO_ADDR register) should be executed to ensure the last clock change instruction gets written to the CCFIFO.

Offset: 0x3e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	CCFIFO_ADDR: clock change FIFO address register.

20.11.2.149 EMC_CCFIFO_DATA_0

CCFIFO_DATA: This register contains the 32-bit data of the EMC register that is intended to be written to or pointed at by CCFIFO_ADDR.

Offset: 0x3ec | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	CCFIFO_DATA: clock change FIFO data register.

20.11.2.150 EMC_CCFIFO_STATUS_0

Offset: 0x3f0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5:0	X	CCFIFO_COUNT

20.11.2.151 EMC_CDB_CNTL_1_0

CDB Control Register

Note: The controlling of CDB phase select is by the CDB_CNTL_1 register of each EMC0/1. The controlling of CDB clock gating is by EMC0's CDB_CNTL_2 register only. The EMC1 version of the CDB_CNTL_2 register is not used.

The following table lists the mapping for the SEL_PI_TAP bits in CDB Control Register 1.

Table 52: SEL_PI_TAP Mapping

CDB SEL_PI_TAP#	Which EMC?	Comment
CDB SEL_PI_TAP15	EMC0_EMC2PMACRO_SEL_PI_TAP7	channel 0, data byte 3
CDB SEL_PI_TAP14	EMC0_EMC2PMACRO_SEL_PI_TAP6	channel 0, data byte 1
CDB SEL_PI_TAP13	EMC0_EMC2PMACRO_SEL_PI_TAP5	channel 0, data byte 0
CDB SEL_PI_TAP12	EMC0_EMC2PMACRO_SEL_PI_TAP4	channel 0, data byte 2
CDB SEL_PI_TAP11	EMC0_EMC2PMACRO_SEL_PI_TAP3	channel 0, BA[2:0], RESET, MA[13:10,3:0]
CDB SEL_PI_TAP10	EMC0_EMC2PMACRO_SEL_PI_TAP2	channel 0, CLK
CDB SEL_PI_TAP9	EMC0_EMC2PMACRO_SEL_PI_TAP1	channel 0, CKE[1:0], CS[1:0], ODT[1:0]
CDB SEL_PI_TAP8	EMC0_EMC2PMACRO_SEL_PI_TAP0	channel 0, MA[14:9:4], RAS, CAS, WE
CDB SEL_PI_TAP7	EMC1_EMC2PMACRO_SEL_PI_TAP7	channel 1, BA[2:0], MA[13:10,3:0]
CDB SEL_PI_TAP6	EMC1_EMC2PMACRO_SEL_PI_TAP6	channel 1, CKE[1:0], CS[1:0], ODT[1:0]
CDB SEL_PI_TAP5	EMC1_EMC2PMACRO_SEL_PI_TAP5	channel 1, CLK
CDB SEL_PI_TAP4	EMC1_EMC2PMACRO_SEL_PI_TAP4	channel 1, MA[14:9:4], RAS, CAS, WE
CDB SEL_PI_TAP3	EMC1_EMC2PMACRO_SEL_PI_TAP3	channel 1, data byte 3
CDB SEL_PI_TAP2	EMC1_EMC2PMACRO_SEL_PI_TAP2	channel 1, data byte 1
CDB SEL_PI_TAP1	EMC1_EMC2PMACRO_SEL_PI_TAP1	channel 1, data byte 0
CDB SEL_PI_TAP0	EMC1_EMC2PMACRO_SEL_PI_TAP0	channel 1, data byte 2

Offset: 0x3f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00x00x00x00x00x00x00x00)

Bit	Reset	Description
22:21	0x0	EMC2PMACRO_SEL_PI_TAP7: [PMC]
19:18	0x0	EMC2PMACRO_SEL_PI_TAP6: [PMC]
16:15	0x0	EMC2PMACRO_SEL_PI_TAP5: [PMC]
13:12	0x0	EMC2PMACRO_SEL_PI_TAP4: [PMC]
10:9	0x0	EMC2PMACRO_SEL_PI_TAP3: [PMC]
7:6	0x0	EMC2PMACRO_SEL_PI_TAP2: [PMC]
4:3	0x0	EMC2PMACRO_SEL_PI_TAP1: [PMC]
1:0	0x0	EMC2PMACRO_SEL_PI_TAP0: [PMC]

20.11.2.152 EMC_CDB_CNTL_2_0

Offset: 0x3f8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	EMC2PMACRO_CLKNIGATE_TAP15: [PMC]
30	0x0	EMC2PMACRO_CLKNIGATE_TAP14: [PMC]
29	0x0	EMC2PMACRO_CLKNIGATE_TAP13: [PMC]
28	0x0	EMC2PMACRO_CLKNIGATE_TAP12: [PMC]
27	0x0	EMC2PMACRO_CLKNIGATE_TAP11: [PMC]
26	0x0	EMC2PMACRO_CLKNIGATE_TAP10: [PMC]
25	0x0	EMC2PMACRO_CLKNIGATE_TAP9: [PMC]
24	0x0	EMC2PMACRO_CLKNIGATE_TAP8: [PMC]
23	0x0	EMC2PMACRO_CLKNIGATE_TAP7: [PMC]
22	0x0	EMC2PMACRO_CLKNIGATE_TAP6: [PMC]
21	0x0	EMC2PMACRO_CLKNIGATE_TAP5: [PMC]
20	0x0	EMC2PMACRO_CLKNIGATE_TAP4: [PMC]
19	0x0	EMC2PMACRO_CLKNIGATE_TAP3: [PMC]
18	0x0	EMC2PMACRO_CLKNIGATE_TAP2: [PMC]
17	0x0	EMC2PMACRO_CLKNIGATE_TAP1: [PMC]
16	0x0	EMC2PMACRO_CLKNIGATE_TAP0: [PMC]
15	0x0	EMC2PMACRO_CLKPIGATE_TAP15: [PMC]
14	0x0	EMC2PMACRO_CLKPIGATE_TAP14: [PMC]
13	0x0	EMC2PMACRO_CLKPIGATE_TAP13: [PMC]
12	0x0	EMC2PMACRO_CLKPIGATE_TAP12: [PMC]
11	0x0	EMC2PMACRO_CLKPIGATE_TAP11: [PMC]
10	0x0	EMC2PMACRO_CLKPIGATE_TAP10: [PMC]
9	0x0	EMC2PMACRO_CLKPIGATE_TAP9: [PMC]
8	0x0	EMC2PMACRO_CLKPIGATE_TAP8: [PMC]
7	0x0	EMC2PMACRO_CLKPIGATE_TAP7: [PMC]
6	0x0	EMC2PMACRO_CLKPIGATE_TAP6: [PMC]
5	0x0	EMC2PMACRO_CLKPIGATE_TAP5: [PMC]
4	0x0	EMC2PMACRO_CLKPIGATE_TAP4: [PMC]
3	0x0	EMC2PMACRO_CLKPIGATE_TAP3: [PMC]
2	0x0	EMC2PMACRO_CLKPIGATE_TAP2: [PMC]
1	0x0	EMC2PMACRO_CLKPIGATE_TAP1: [PMC]
0	0x0	EMC2PMACRO_CLKPIGATE_TAP0: [PMC]

20.11.2.153 EMC_XM2CLKPADCTRL2_0

XM2CLK Pad Control Register

Boot requirements:

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x3fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx00000xxx00000)

Bit	Reset	Description
12:8	0x0	EMC2TMC_CFG_XM2CLK_DLY_TRM_P: [PMC]
4:0	0x0	EMC2TMC_CFG_XM2CLK_DLY_TRM_N: [PMC]

20.11.2.154 EMC_SWIZZLE_RANK0_BYTE_CFG_0

For byte swizzling, each SWZ_RANK*_BYTE*_SEL field indicates which SDRAM byte is mapped/connected to the corresponding byte of the chip.

The SWZ_RANK*_BYTE*_BIT*_SEL fields indicate which SDRAM data bit (0-7) within each byte is mapped/connected to the corresponding bit of the chip.

Offset: 0x400 | Read/Write: R/W | Reset: 0x000000e4 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx11100100)

Bit	Reset	Description
7:6	0x3	SWZ_RANK0_BYTE3_SEL: [PMC]
5:4	0x2	SWZ_RANK0_BYTE2_SEL: [PMC]
3:2	0x1	SWZ_RANK0_BYTE1_SEL: [PMC]
1:0	0x0	SWZ_RANK0_BYTE0_SEL: [PMC]

20.11.2.155 EMC_SWIZZLE_RANK0_BYTE0_0

Offset: 0x404 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK0_BYTE0_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK0_BYTE0_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK0_BYTE0_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK0_BYTE0_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK0_BYTE0_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK0_BYTE0_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK0_BYTE0_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK0_BYTE0_BIT0_SEL: [PMC]

20.11.2.156 EMC_SWIZZLE_RANK0_BYTE1_0

Offset: 0x408 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK0_BYTE1_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK0_BYTE1_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK0_BYTE1_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK0_BYTE1_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK0_BYTE1_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK0_BYTE1_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK0_BYTE1_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK0_BYTE1_BIT0_SEL: [PMC]

20.11.2.157 EMC_SWIZZLE_RANK0_BYTE2_0

Offset: 0x40c | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK0_BYTE2_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK0_BYTE2_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK0_BYTE2_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK0_BYTE2_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK0_BYTE2_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK0_BYTE2_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK0_BYTE2_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK0_BYTE2_BIT0_SEL: [PMC]

20.11.2.158 EMC_SWIZZLE_RANK0_BYTE3_0

Offset: 0x410 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK0_BYTE3_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK0_BYTE3_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK0_BYTE3_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK0_BYTE3_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK0_BYTE3_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK0_BYTE3_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK0_BYTE3_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK0_BYTE3_BIT0_SEL: [PMC]

20.11.2.159 EMC_SWIZZLE_RANK1_BYTE_CFG_0

Offset: 0x414 | Read/Write: R/W | Reset: 0x000000e4 (0bxxxxxxxxxxxxxxxxxxxxxxxxx11100100)

Bit	Reset	Description
7:6	0x3	SWZ_RANK1_BYTE3_SEL: [PMC]
5:4	0x2	SWZ_RANK1_BYTE2_SEL: [PMC]
3:2	0x1	SWZ_RANK1_BYTE1_SEL: [PMC]
1:0	0x0	SWZ_RANK1_BYTE0_SEL: [PMC]

20.11.2.160 EMC_SWIZZLE_RANK1_BYTE0_0

Offset: 0x418 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK1_BYTE0_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK1_BYTE0_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK1_BYTE0_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK1_BYTE0_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK1_BYTE0_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK1_BYTE0_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK1_BYTE0_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK1_BYTE0_BIT0_SEL: [PMC]

20.11.2.161 EMC_SWIZZLE_RANK1_BYTE1_0

Offset: 0x41c | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK1_BYTE1_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK1_BYTE1_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK1_BYTE1_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK1_BYTE1_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK1_BYTE1_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK1_BYTE1_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK1_BYTE1_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK1_BYTE1_BIT0_SEL: [PMC]

20.11.2.162 EMC_SWIZZLE_RANK1_BYTE2_0

Offset: 0x420 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK1_BYTE2_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK1_BYTE2_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK1_BYTE2_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK1_BYTE2_BIT4_SEL: [PMC]

Bit	Reset	Description
14:12	0x3	SWZ_RANK1_BYTE2_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK1_BYTE2_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK1_BYTE2_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK1_BYTE2_BIT0_SEL: [PMC]

20.11.2.163 EMC_SWIZZLE_RANK1_BYTE3_0

Offset: 0x424 | Read/Write: R/W | Reset: 0x76543210 (0bx111x110x101x100x011x010x001x000)

Bit	Reset	Description
30:28	0x7	SWZ_RANK1_BYTE3_BIT7_SEL: (encoded)
26:24	0x6	SWZ_RANK1_BYTE3_BIT6_SEL: (encoded)
22:20	0x5	SWZ_RANK1_BYTE3_BIT5_SEL: [PMC]
18:16	0x4	SWZ_RANK1_BYTE3_BIT4_SEL: [PMC]
14:12	0x3	SWZ_RANK1_BYTE3_BIT3_SEL: [PMC]
10:8	0x2	SWZ_RANK1_BYTE3_BIT2_SEL: [PMC]
6:4	0x1	SWZ_RANK1_BYTE3_BIT1_SEL: [PMC]
2:0	0x0	SWZ_RANK1_BYTE3_BIT0_SEL: [PMC]

20.11.2.164 EMC_CA_TRAINING_START_0

Procedure for LPDDR3 CA Training

(1a) Configure data swizzling (if needed) through the DATA_SWIZZLE_RANK[0,1]_[0:7] registers.

(1b) Enable EMC2PMACRO_CFG_BYPASS_ADDRPIPE if needed. There's no need to enable EMC2PMACRO_CFG_BYPASS_DATAPIPE* for CA training purposes.

(1c) Follow the procedure on optimal trim setting into the CA bus clock trimmer below.

(2) Configure the CA training engine through the CA_TRAINING_[TIMING_CNTL1,TIMING_CNTL2, CA_LEAD_IN, CA, and CA_LEAD_OUT] registers.

(3) Software configures the CA training engine to use MR41 mode (through the CA_TRAINING_CFG register).

(4) Software starts the CA training engine (in the CA_TRAINING_START register) and waits for the CA training to finish (CA_TRAINING_BUSY).

(5) Software saves the result from the CA_TRAINING_RESULT* registers.

(6) Software repeats steps 3-5 but configures for MR48 mode.

(7) Software combines the results from MR41/MR48 modes to come up with a single trim setting and programs the CA bus clock trimmer (see below).

(8) Software issues MR42 to rank0.

(9) If desired, software can repeat steps 3-8 for rank1.

(10) Software combines the results from MR41/MR48 modes (if desired, results for rank1 as well) to come up with a single trim setting and programs the CA bus clock trimmer (see below).

(11) A separate CA training needs to be performed on channel-1 as well. Theoretically, CA training can be performed on both channels 0/1 in parallel to save training time.

Procedure to program the optimal trim setting into the CA bus clock trimmer

(A) Put CLKBUF0/CLKBUF2 into trimmer mode by setting EMC2TMC_CFG_XM2CMD_CLKBUF[0,2]_CONFIG = 001.

(B) Put XFORM_ADDR[0,2]_MULT field to 5'b0.

(C) Program bits [7:2] in the XFORM_ADDR[0,2]_OFFS field (11 bits) with the optimal trim result. The other XFORM_ADDR[0,2]_OFFS bits should remain at 0.

(D) Write a 1 into the TIMING_UPDATE field to load the above fields from shadow to active.

Below is what should be programmed into the CA_TRAINING_TIMING_CONTROL* registers.

CACKEL = tCACKEL + 15 clks

CAENT = tCAENT + 3 clks

CACD = tCACD + 5 clks

CACKEH = tCACKEH – 1 clk

CAEXT = tCAEXT – 1 clk

CA_DQ_RDV = CACD

Offset: 0x428 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	CA_TRAIN_START: 1 = start training.

20.11.2.165 EMC_CA_TRAINING_BUSY_0

Offset: 0x42c | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	CA_TRAIN_BUSY: 1 = training is in progress.

20.11.2.166 EMC_CA_TRAINING_CFG_0

Offset: 0x430 | Read/Write: R/W | Reset: 0x0000001f (0bx000xx0000000000x0000000x0011111)

Bit	Reset	Description
30	RANK0	CA_RANK: Indicates which rank to do training on. 0 = RANK0 1 = RANK1
29	0x0	CA_DRAM_X32: 1 = Using x32 DRAM, 0 = using x16 DRAM.
28	MR41	CA_MR48: 0 = training with MR41 mode, 1 = training with MR48 mode. 0 = MR41 1 = MR48
25:16	0x0	CA_MRW_CA: MR41 = 0x290, MR48 = 0x300.
14:8	0x0	CA_START_TRIM_VAL: Indicates the starting delay trim value to use for CA training.
6:0	0x1f	CA_END_TRIM_VAL: Indicates the ending delay trim value to use for CA training.

20.11.2.167 EMC_CA_TRAINING_TIMING_CNTL1_0

Offset: 0x434 | Read/Write: R/W | Reset: 0x1f7df7df (0bxxx11111x11111x11111x11111x11111)

Bit	Reset	Description
28:24	0x1f	CAEXT: tCAEXT parameter.
22:18	0x1f	CACKEH: tCACKEH parameter.
16:12	0x1f	CACD: tCACD parameter.
10:6	0x1f	CAENT: tCAENT parameter.
4:0	0x1f	CACKEL: tCACKEL parameter.

20.11.2.168 EMC_CA_TRAINING_TIMING_CNTL2_0

Offset: 0x438 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx11111)

Bit	Reset	Description
4:0	0x1f	CA_DQ_RDV: number of clocks to delay after CS is asserted before comparing CA/DQ. Minimum is 5.

20.11.2.169 EMC_CA_TRAINING_CA_LEAD_IN_0

Offset: 0x43c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx0000000000xxxxxx0000000000)

Bit	Reset	Description
25:16	0x0	CA_LEAD_IN_RISE: CA rise value for the clock before CS is active.
9:0	0x0	CA_LEAD_IN_FALL: CA fall value for the clock before CS is active.

20.11.2.170 EMC_CA_TRAINING_CA_0

Offset: 0x440 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx0000000000xxxxxx0000000000)

Bit	Reset	Description
25:16	0x0	CA_RISE: CA rise value for the clock CS is active.
9:0	0x0	CA_FALL: CA fall value for the clock CS is active.

20.11.2.171 EMC_CA_TRAINING_CA_LEAD_OUT_0

Offset: 0x444 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx0000000000xxxxxx0000000000)

Bit	Reset	Description
25:16	0x0	CA_LEAD_OUT_RISE: CA rise value for the clock after CS is active.
9:0	0x0	CA_LEAD_OUT_FALL: CA fall value for the clock after CS is active.

20.11.2.172 EMC_CA_TRAINING_RESULT1_0

Offset: 0x448 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CA_TRIM_RESULT_31_0: Indicates which trim setting(s) pass the CA-to-DQ comparison.

20.11.2.173 EMC_CA_TRAINING_RESULT2_0

Offset: 0x44c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CA_TRIM_RESULT_63_32: Indicates which trim setting(s) pass the CA-to-DQ comparison.

20.11.2.174 EMC_CA_TRAINING_RESULT3_0

Offset: 0x450 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CA_TRIM_RESULT_95_64: Indicates which trim setting(s) pass the CA-to-DQ comparison.

20.11.2.175 EMC_CA_TRAINING_RESULT4_0

Offset: 0x454 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CA_TRIM_RESULT_127_96: Indicates which trim setting(s) pass the CA-to-DQ comparison.

20.11.2.176 EMC_AUTO_CAL_CONFIG2_0

Auto-Calibration Settings for EMC Pads

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x458 | Read/Write: R/W | Reset: 0x00000000 (0bxxx00000xxx00000xxx00000xxx00000)

Bit	Reset	Description
28:24	0x0	AUTO_CAL_CNTL_PD_OFFSET: [PMC] 2's complement offset for CS/ODT/CKE pull-down value
20:16	0x0	AUTO_CAL_CNTL_PU_OFFSET: [PMC] 2's complement offset for CS/ODT/CKE pull-up value
12:8	0x0	AUTO_CAL_CMD_PD_OFFSET: [PMC] 2's complement offset for ADDR/RAS/CAS/WE pull-down value
4:0	0x0	AUTO_CAL_CMD_PU_OFFSET: [PMC] 2's complement offset for ADDR/RAS/CAS/WE pull-up value

20.11.2.177 EMC_AUTO_CAL_CONFIG3_0

Auto-Calibration Settings for EMC Pads

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 0x45c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx00000xxx00000)

Bit	Reset	Description
12:8	0x0	AUTO_CAL_CLK_PD_OFFSET: [PMC] 2's complement offset for CLK pull-down value
4:0	0x0	AUTO_CAL_CLK_PU_OFFSET: [PMC] 2's complement offset for CLK pull-up value

20.11.2.178 EMC_AUTO_CAL_STATUS2_0

EMC Pad Calibration Status

Offset: 0x460 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:24	X	AUTO_CAL_CNTL_PULLDOWN_ADJ: Pull-down code sent to pads
20:16	X	AUTO_CAL_CNTL_PULLUP_ADJ: Pull-up code sent to pads
12:8	X	AUTO_CAL_CMD_PULLDOWN_ADJ: Pull-down code sent to pads.
4:0	X	AUTO_CAL_CMD_PULLUP_ADJ: Pull-up code sent to pads

20.11.2.179 EMC_XM2CMDPADCTRL3_0

Offset: 0x464 | Read/Write: R/W | Reset: 0x050c0000 (0b00000101000011000xxxxxxxxxxxxx) | Default: 0x858c0000

Bit	Reset	SW Default	Description
31:28	0x0	0x8	EMC2PMACRO_CFG_XM2CNTL_DRVUP_SLWF: [PMC]
27:24	0x5	0x5	EMC2PMACRO_CFG_XM2CNTL_DRVDN_SLWR: [PMC]
23:19	0x1	0x11	EMC2PMACRO_CFG_XM2CNTL_DRVUP: [PMC]
18:14	0x10	0x10	EMC2PMACRO_CFG_XM2CNTL_DRVDN: [PMC]

20.11.2.180 EMC_IBDLY_0

DRAM timing parameter

Offset: 0x468 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00001)

Bit	Reset	Description
4:0	0x1	IBDLY: [PMC] tells the chip when to change IBDLY for DQ/DQS.

20.11.2.181 EMC_DLL_XFORM_ADDR0_0

Configure Digital DLL for CMD/ADDR Group 0

XFORM_ADDRx_MULT and XFORM_ADDRx_OFFS are a multiplier and offset, respectively, that can be used to modify the DLL output. Final products may be read out in the ADDR_TRIMMER_RD register. The default is multiply by 1, to keep the DLL's 1/4 cycle delay. Integer + fractional formats for multiplier and offset:

OFFS is 2's complement format, with bit [0] representing integer

The output of the XFORM is xform_out = (xform_in * MULT + OFFS*16)/16

The largest allowed MULT = 0x17

In the case of overflow, the output is clamped according to these rules:

out >= 0x600, xform_out = 0x000

0x600 > out >= 0x400, xform_out = 0x3ff

That means clamping works within 50% above the maximum (0x3ff) DLL output or 50% below the minimum (0) DLL output.

To make sure that is always satisfied, conservatively program OFFS so that 0x600 < OFFS < (0x5ff - ((MULT * 0x3ff) >> 4))

The DLL output is effectively overridden by setting XFORM_*_MULT to 0 and programming XFORM_*_OFFS[8:1] to override the value

This register is shadowed: see usage note at the top.

Boot requirements

- This register should be parameterized in the BCT and written by the Boot ROM during cold boot.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

The ADDR(0|1|2)_CURRENT_TRIM_VAL registers provide a way to observe the values being used by the trimmers.

Offset: 0x46c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000000000xxxxxxx00000)

Bit	Reset	Description
22:12	0x0	XFORM_ADDR0_OFFS: [PMC]
4:0	0x0	XFORM_ADDR0_MULT: [PMC]

20.11.2.182 EMC_DLL_XFORM_ADDR1_0

Configure Digital DLL for CMD/ADDR Group 1

Offset: 0x470 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000000000xxxxxxx00000)

Bit	Reset	Description
22:12	0x0	XFORM_ADDR1_OFFS: [PMC]
4:0	0x0	XFORM_ADDR1_MULT: [PMC]

20.11.2.183 EMC_DLL_XFORM_ADDR2_0

Configure Digital DLL for CMD/ADDR Group 2

Offset: 0x474 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx00000000000xxxxxxx00000)

Bit	Reset	Description
22:12	0x0	XFORM_ADDR2_OFFS: [PMC]
4:0	0x0	XFORM_ADDR2_MULT: [PMC]

20.11.2.184 EMC_DLI_ADDR_TRIM_0

Configure Digital DLL

Offset: 0x478 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:20	X	ADDR2_CURRENT_TRIM_VAL
19:10	X	ADDR1_CURRENT_TRIM_VAL
9:0	X	ADDR0_CURRENT_TRIM_VAL

20.11.2.185 EMC_DSR_VTTGEN_DRV_0

Offset: 0x47c | Read/Write: R/W | Reset: 0x0000003f (0bxxxxx000xxxxx000xxxxxxxxx111111)

Bit	Reset	Description
26:24	0x0	DSR_VTTGEN_DRVUP: [PMC] Indicates the VTTGEN VCLAMP regulator impedance to use during DSR.
18:16	0x0	DSR_VTTGEN_DRVDN: [PMC] Indicates the VTTGEN VAUXP regulator impedance to use during DSR.
5:0	0x3f	DSR_VTTGEN_E_NO_VTTGEN: [PMC] Disables optional VTTGEN pads (1 bit per pad) [0]--ADDR0, [1]--ADDR1, [2]--ADDR2, [3]--DATA1, [4]--DATA2, [5]--DATA3

20.11.2.186 EMC_TXDSRVTTGEN_0

DRAM timing parameter

Offset: 0x480 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	TXDSRVTTGEN: [PMC] Cycles to wait from DSR exit (VTTGEN drive normal), to when a DRAM transaction is allow to start.

20.11.2.187 EMC_XM2CMDPADCTRL4_0

Controls related to NI-PI handoff in pad macros.

Offset: 0x484 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx0x0x0x0x0x0)

Bit	Reset	Description
10	0x0	EMC2PMACRO_CFG_2T_PM_PHASE_SHIFT: [PMC] when set to 1, enables 1/2 clks of delay on OB 2T signals
8	0x0	EMC2PMACRO_CFG_2T_PM2PAD_FLOP_BYPASS: [PMC] when set to 1, enables flop between NI-PI handoff structure and pad for OB 2T
6	0x0	EMC2PMACRO_CFG_1T_PM_PHASE_SHIFT: [PMC] when set to 1, enables 1/2 clks of delay on OB 1T signals
4	0x0	EMC2PMACRO_CFG_1T_PM2PAD_FLOP_BYPASS: [PMC] when set to 1, enables flop between NI-PI handoff structure and pad for OB 1T
2	0x0	EMC2PMACRO_CFG_DAT_PM_PHASE_SHIFT: [PMC] when set to 1, enables 1/2 clks of delay on OB data signals
0	0x0	EMC2PMACRO_CFG_DAT_PM2PAD_FLOP_BYPASS: [PMC] when set to 1, enables flop between DQ/DQS NI-PI handoff structure and pad

20.11.2.188 EMC_ADDR_SWIZZLE_STACK1A_0

Address swizzling of the 3 ADDR/CNTL I/O stacks:

- Address stack1 (12 signals): BA[2:0], ADR[3:0], ADR[13:10], RESETN.
- Address stack2 (10 signals): ADR[9:4], ADR[14], CAS, RAS, WE.
- Address stack3 (6 signals): CS[1:0], CKE[1:0], ODT[1:0].

Note: The controlling of CDB phase select is by the CDB_CNTL_1 register of each EMC0/1.
The controlling of CDB clock gating is by EMC0's CDB_CNTL_2 register only. The EMC1 version of the CDB_CNTL_2 register is not used.
If LPDDRX_ADR_PIN_SWIZZLE_EN is enabled, it will get swizzled first before the ADDR/CNTL I/O stack swizzling.
For unused pins (such as BA0 pin in LPDDR2 mode), programming to anything other than the allowed enum will force that pin to be inactive or not-toggle

Offset: 0x488 | Read/Write: R/W | Reset: 0x65430210 (0b0110010101000011xxxx001000010000)

Bit	Reset	Description
31:28	ADR3	SWZ_STACK1_ADR3:[PMC_SECURE] 0 = BA0 1 = BA1 2 = BA2 3 = ADR0 4 = ADR1 5 = ADR2 6 = ADR3 7 = ADR10 8 = ADR11 9 = ADR12 10 = ADR13 11 = RESETN
27:24	ADR2	SWZ_STACK1_ADR2:[PMC_SECURE] 0 = BA0 1 = BA1 2 = BA2 3 = ADR0 4 = ADR1 5 = ADR2 6 = ADR3 7 = ADR10 8 = ADR11 9 = ADR12 10 = ADR13 11 = RESETN
23:20	ADR1	SWZ_STACK1_ADR1:[PMC_SECURE] 0 = BA0 1 = BA1 2 = BA2 3 = ADR0 4 = ADR1 5 = ADR2 6 = ADR3 7 = ADR10 8 = ADR11 9 = ADR12 10 = ADR13 11 = RESETN

Bit	Reset	Description
19:16	ADR0	SWZ_STACK1_ADR0:[PMC_SECURE] 0 = BA0 1 = BA1 2 = BA2 3 = ADR0 4 = ADR1 5 = ADR2 6 = ADR3 7 = ADR10 8 = ADR11 9 = ADR12 10 = ADR13 11 = RESETN
11:8	BA2	SWZ_STACK1_BA2:[PMC_SECURE] 0 = BA0 1 = BA1 2 = BA2 3 = ADR0 4 = ADR1 5 = ADR2 6 = ADR3 7 = ADR10 8 = ADR11 9 = ADR12 10 = ADR13 11 = RESETN
7:4	BA1	SWZ_STACK1_BA1:[PMC_SECURE] 0 = BA0 1 = BA1 2 = BA2 3 = ADR0 4 = ADR1 5 = ADR2 6 = ADR3 7 = ADR10 8 = ADR11 9 = ADR12 10 = ADR13 11 = RESETN
3:0	BA0	SWZ_STACK1_BA0:[PMC_SECURE] 0 = BA0 1 = BA1 2 = BA2 3 = ADR0 4 = ADR1 5 = ADR2 6 = ADR3 7 = ADR10 8 = ADR11 9 = ADR12 10 = ADR13 11 = RESETN

20.11.2.189 EMC_ADDR_SWIZZLE_STACK1B_0

Offset: 0x48c | Read/Write: R/W | Reset: 0x000ba987 (0bxxxxxxxxxxxxx10111010100110000111)

Bit	Reset	Description
19:16	RESETN	SWZ_STACK1_RESETN:[PMC_SECURE] 0 = BA0 1 = BA1 2 = BA2 3 = ADR0 4 = ADR1

Bit	Reset	Description
		5 = ADR2 6 = ADR3 7 = ADR10 8 = ADR11 9 = ADR12 10 = ADR13 11 = RESETN
15:12	ADR13	SWZ_STACK1_ADR13:[PMC_SECURE] 0 = BA0 1 = BA1 2 = BA2 3 = ADR0 4 = ADR1 5 = ADR2 6 = ADR3 7 = ADR10 8 = ADR11 9 = ADR12 10 = ADR13 11 = RESETN
11:8	ADR12	SWZ_STACK1_ADR12:[PMC_SECURE] 0 = BA0 1 = BA1 2 = BA2 3 = ADR0 4 = ADR1 5 = ADR2 6 = ADR3 7 = ADR10 8 = ADR11 9 = ADR12 10 = ADR13 11 = RESETN
7:4	ADR11	SWZ_STACK1_ADR11:[PMC_SECURE] 0 = BA0 1 = BA1 2 = BA2 3 = ADR0 4 = ADR1 5 = ADR2 6 = ADR3 7 = ADR10 8 = ADR11 9 = ADR12 10 = ADR13 11 = RESETN
3:0	ADR10	SWZ_STACK1_ADR10:[PMC_SECURE] 0 = BA0 1 = BA1 2 = BA2 3 = ADR0 4 = ADR1 5 = ADR2 6 = ADR3 7 = ADR10 8 = ADR11 9 = ADR12 10 = ADR13 11 = RESETN

20.11.2.190 EMC_ADDR_SWIZZLE_STACK2A_0

Offset: 0x490 | Read/Write: R/W | Reset: 0x06543210 (0bxxxx01100101010000110010000010000)

Bit	Reset	Description
27:24	ADR14	SWZ_STACK2_ADR14:[PMC_SECURE] 0 = ADR4 1 = ADR5 2 = ADR6 3 = ADR7 4 = ADR8 5 = ADR9 6 = ADR14 7 = CAS 8 = RAS 9 = WE
23:20	ADR9	SWZ_STACK2_ADR9:[PMC_SECURE] 0 = ADR4 1 = ADR5 2 = ADR6 3 = ADR7 4 = ADR8 5 = ADR9 6 = ADR14 7 = CAS 8 = RAS 9 = WE
19:16	ADR8	SWZ_STACK2_ADR8:[PMC_SECURE] 0 = ADR4 1 = ADR5 2 = ADR6 3 = ADR7 4 = ADR8 5 = ADR9 6 = ADR14 7 = CAS 8 = RAS 9 = WE
15:12	ADR7	SWZ_STACK2_ADR7:[PMC_SECURE] 0 = ADR4 1 = ADR5 2 = ADR6 3 = ADR7 4 = ADR8 5 = ADR9 6 = ADR14 7 = CAS 8 = RAS 9 = WE
11:8	ADR6	SWZ_STACK2_ADR6:[PMC_SECURE] 0 = ADR4 1 = ADR5 2 = ADR6 3 = ADR7 4 = ADR8 5 = ADR9 6 = ADR14 7 = CAS 8 = RAS 9 = WE
7:4	ADR5	SWZ_STACK2_ADR5:[PMC_SECURE] 0 = ADR4 1 = ADR5 2 = ADR6 3 = ADR7 4 = ADR8

Bit	Reset	Description
		5 = ADR9 6 = ADR14 7 = CAS 8 = RAS 9 = WE
3:0	ADR4	SWZ_STACK2_ADR4:[PMC_SECURE] 0 = ADR4 1 = ADR5 2 = ADR6 3 = ADR7 4 = ADR8 5 = ADR9 6 = ADR14 7 = CAS 8 = RAS 9 = WE

20.11.2.191 EMC_ADDR_SWIZZLE_STACK2B_0

Offset: 0x494 | Read/Write: R/W | Reset: 0x00000987 (0bxxxxxxxxxxxxxxxxxxxxx100110000111)

Bit	Reset	Description
11:8	WE	SWZ_STACK2_WE:[PMC_SECURE] 0 = ADR4 1 = ADR5 2 = ADR6 3 = ADR7 4 = ADR8 5 = ADR9 6 = ADR14 7 = CAS 8 = RAS 9 = WE
7:4	RAS	SWZ_STACK2_RAS:[PMC_SECURE] 0 = ADR4 1 = ADR5 2 = ADR6 3 = ADR7 4 = ADR8 5 = ADR9 6 = ADR14 7 = CAS 8 = RAS 9 = WE
3:0	CAS	SWZ_STACK2_CAS:[PMC_SECURE] 0 = ADR4 1 = ADR5 2 = ADR6 3 = ADR7 4 = ADR8 5 = ADR9 6 = ADR14 7 = CAS 8 = RAS 9 = WE

20.11.2.192 EMC_ADDR_SWIZZLE_STACK3_0

Offset: 0x498 | Read/Write: R/W | Reset: 0x00543210 (0bxxxxxxxxx101x100x011x010x001x000)

Bit	Reset	Description
22:20	ODT1	SWZ_STACK3_ODT1:[PMC_SECURE] 0 = CS0 1 = CS1 2 = CKE0 3 = CKE1 4 = ODT0 5 = ODT1
18:16	ODT0	SWZ_STACK3_ODT0:[PMC_SECURE] 0 = CS0 1 = CS1 2 = CKE0 3 = CKE1 4 = ODT0 5 = ODT1
14:12	CKE1	SWZ_STACK3_CKE1:[PMC_SECURE] 0 = CS0 1 = CS1 2 = CKE0 3 = CKE1 4 = ODT0 5 = ODT1
10:8	CKE0	SWZ_STACK3_CKE0:[PMC_SECURE] 0 = CS0 1 = CS1 2 = CKE0 3 = CKE1 4 = ODT0 5 = ODT1
6:4	CS1	SWZ_STACK3_CS1:[PMC_SECURE] 0 = CS0 1 = CS1 2 = CKE0 3 = CKE1 4 = ODT0 5 = ODT1
2:0	CS0	SWZ_STACK3_CS0:[PMC_SECURE] 0 = CS0 1 = CS1 2 = CKE0 3 = CKE1 4 = ODT0 5 = ODT1

21.0 AHB

21.1 AHB Bus

The AHB Bus conforms to the *AMBA® Specification (Rev 2.0) Advanced High-performance Bus (AHB)* architecture as published by ARM®.

AHB is a 32-bit multi-master bus. Despite what its name implies, it is a second tier bus in the Tegra® 4 processor, slower and less flexible than the AXI bus used by the CPU, or the memory client interface used by the high-speed devices.

AHB clients in Tegra 4 processors include:

- The AHB memory controller slave, which is the interface between the AHB bus and the Memory Controller
- The AVP crossbar, both as a master and as a slave. This is the path to IRAM from AHB devices, and also the AVP and CPU path to AHB devices. Neither the CPU nor the AVP can access DRAM through this path.
- The APB DMA controller, used to provide data transfer between APB devices and memory.
- The USB2-OTG controllers
- The NAND flash controller
- The NOR/MIO controller
- The SD/MMC controllers
- The MIPI HSI controller
- The security engine (SE), both as a master and as a slave
- BSEA and BSEV bitstream engines
- The CoreSight™ debug controller.
- The deprecated AHB DMA controller master

AHB in the Tegra 4 processor supports secure access to memory, using the same secure bit mechanism used by the ARM Cortex™-A15 TrustZone security mechanism. This is supported by the SE, which can make secure accesses to memory.

21.2 AHB Bus Arbiter

The AHB Arbiter controls AHB bus master arbitration. This effectively forms a second level of arbitration for access to the memory controller through the AHB Slave Memory device, although AHB masters in some circumstances can use other AHB slaves, in particular they can access IRAM.

The controls presented here are largely for diagnostic purposes only. For suspect AHB performance problems, try disabling the other masters for the device with performance issues. In normal operation bus parking is usually enabled, and all the masters are enabled.

Also see the AHB slave interface registers, where the AHB pre-fetch logic can be configured to enhance performance for devices doing sequential read access from DRAM.

Each AHB master is assigned to either the high or low priority bin.

21.2.1 Arbitration Scheme

In every AHB Arbitration cycle, it is first decided whether a request from the high or low priority bin will be served. If there are only requests active for one of the bins, then that bin is served. When there are active requests from both bins, then the value from the AHB_PRIORITY_WEIGHT field is considered. There is a counter that is loaded with the AHB_PRIORITY_WEIGHT whenever the current count is 0 and someone wins an AHB arbitration. This counter is decremented anytime the count is

nonzero and the winner of AHB arbitration was from the high-priority bin. Whenever this counter is nonzero, then a high-priority bin request will win over any low-priority bin request. In a system where both high- and low- priority bin requests are constantly active, the AHB_PRIORITY_WEIGHT says how many high-priority requests will be served for every one low-priority request.

Within each bin, the arbitration algorithm is round robin. For example, after AHB Master 2 wins an arbitration, then Master 3 has precedence for winning the next (followed by Master 4, etc. while Master 2 is last). AHB Master ID's can be seen in the enumeration of AHB_MEM_PREFETCH_CFG* registers' "AHB_MST_ID" field.

21.2.2 AHB Arbiter Registers

Refer to "Reading Register Tables" in the Introduction section for the register table protocol as well as recommendations for accessing registers.

In Tegra 4 and prior generation processors, this location used to hold a configuration register for COP_CACHE, but COP_CACHE has been replaced with AVP_CACHE, which has a region in a different section. For software compatibility, this location is reserved.

21.2.2.1 AHB_ARBITRATION_DISABLE_0

The AHB arbitration control register allows user to tweak arbitration behavior of the AHB arbiter.

- Enable bus parking. This keeps the last serviced AHB master on the bus granted so that it can start another transaction faster. If bus parking is disabled, no AHB master will be able to start a new transaction until the arbitration is done and the master is granted the bus.
- Allows the user to specifically disable an AHB master from arbitrating on the AHB bus.

AHB Arbitration Controller

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxx0000000000000x00000000)

Bit	Reset	Description
31	0x0	DIS_BUS_PARK: 1 = Disable bus parking. 0 = ENABLE 1 = DISABLE
30	0x0	DIS_PENULTIMATE_ARB: 1 = Disable arbitration on the second to last transfer. 0 = ENABLE 1 = DISABLE
21	0x0	MIPIHSI: 1 = Disable MIPIHSI from arbitration. 0 = ENABLE 1 = DISABLE
20	0x0	SDMMC3: 1 = Disable SDMMC3 from arbitration. 0 = ENABLE 1 = DISABLE
19	0x0	SDMMC2: 1 = Disable SDMMC2 from arbitration. 0 = ENABLE 1 = DISABLE
18	0x0	USB2: 1 = Disable USB2 from arbitration. 0 = ENABLE 1 = DISABLE
17	0x0	USB3: 1 = Disable USB3 from arbitration. 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
16	0x0	BSEA: 1 = Disable BSEA from arbitration. 0 = ENABLE 1 = DISABLE
15	0x0	DDS: 1 = Disable DDS from arbitration. 0 = ENABLE 1 = DISABLE
14	0x0	SE: 1 = Disable SE from arbitration.
13	0x0	BSEV: 1 = Disable BSEV from arbitration. 0 = ENABLE 1 = DISABLE
12	0x0	SDMMC4: 1 = Disable SDMMC4 from arbitration. 0 = ENABLE 1 = DISABLE
11	0x0	SNOR: 1 = Disable SNOR from arbitration. 0 = ENABLE 1 = DISABLE
10	0x0	NAND: 1 = Disable NAND from arbitration. 0 = ENABLE 1 = DISABLE
9	0x0	SDMMC1: 1 = Disable SDMMC1 from arbitration. 0 = ENABLE 1 = DISABLE
7	0x0	APBDMA: 1 = Disable APB-DMA from arbitration. 0 = ENABLE 1 = DISABLE
6	0x0	USB: 1 = Disable USB from arbitration. 0 = ENABLE 1 = DISABLE
5	0x0	AHBDMA: 1 = Disable AHB-DMA from arbitration. 0 = ENABLE 1 = DISABLE
4	0x0	ARC: 1 = Disable ARC from arbitration. 0 = ENABLE 1 = DISABLE
3	0x0	CSITE: 1 = Disable CoreSight from arbitration. 0 = ENABLE 1 = DISABLE
2	0x0	VCP: 1 = Disable VCP from arbitration. 0 = ENABLE 1 = DISABLE
1	0x0	COP: 1 = Disable COP from arbitration. 0 = ENABLE 1 = DISABLE
0	0x0	CPU: 1 = Disable CPU from arbitration. 0 = ENABLE 1 = DISABLE

21.2.2.2 AHB_ARBITRATION_PRIORITY_CTRL_0

The AHB arbiter implements a 2-level priority scheme. In the first level, arbitration is determined between the high and low priority group according to the priority weight; the higher the weight, the higher the winning rate of the high priority group.

In the second level, within each of the high/low priority group, arbitration is determined in a round-robin fashion.

AHB Arbitration Priority Control Register

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

Bit	Reset	Description
31:29	0x0	AHB_PRIORITY_WEIGHT: AHB priority weight count. This 3-bit field is used to control the amount of attention (weight) given to the high priority group before switching to the low priority group.
28:0	0x0	AHB_PRIORITY_SELECT: 0 = low priority group.

21.2.2.3 AHB_ARBITRATION_USR_PROTECT_0

USR Protection Register

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	CACHE: Abort on USR mode access to Cache memory space 0 = ABT_DIS 1 = ABT_EN
7	0x0	ROM: Abort on USR mode access to internal ROM memory space 0 = ABT_DIS 1 = ABT_EN
6	0x0	APB: Abort on USR mode access to APB memory space 0 = ABT_DIS 1 = ABT_EN
5	0x0	AHB: Abort on USR mode access to AHB memory space 0 = ABT_DIS 1 = ABT_EN
3	0x0	IRAMD: Abort on USR mode access to iRAMd memory space 0 = ABT_DIS 1 = ABT_EN
2	0x0	IRAMC: Abort on USR mode access to iRAMc memory space 0 = ABT_DIS 1 = ABT_EN
1	0x0	IRAMB: Abort on USR mode access to iRAMb memory space 0 = ABT_DIS 1 = ABT_EN
0	0x0	IRAMA: Abort on USR mode access to iRAMa memory space 0 = ABT_DIS 1 = ABT_EN

21.3 AHB “Gizmo”

AHB master/slave gizmos are essentially hardware layers used by many of the master/slave logic which need to connect to the AHB bus.

These hardware layers handle all the AHB bus protocols and convert the more complex AHB protocol into a much simpler IP interface/handshake.

Because the AHB master/slave gizmos interface with many IP logic with different characteristics varying in speed, latency, etc., the gizmos accept a number of static configuration bits. Depending on system speed, latency requirement, transfer direction, burst characteristic, etc., these static configuration bits can be pre-configured to give extra performance or improve bus efficiency.

21.3.1 AHB Gizmo Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

21.3.1.1 AHB_GIZMO_AHB_MEM_0

AHB Master/Slave Gizmo Register

Given below is a description of each configuration field, what they are used for and the pros and cons of using them.

> **Note:** These configuration bits are meant to be changed only when the particular gizmo you want to change is idle (not being used). Changing the configuration bits on-the-fly while that gizmo is active can hang the system.

AHB gizmo configuration bit definition:

(1) AHB Master Gizmo

1. MAX_AHB_BURSTSIZE: Controls the maximum burst size that this gizmo can generate on the AHB bus. For the current system, this field should be set to burst-of-8.
2. IMMEDIATE: Controls how quickly an AHB write request on the bus can start.
   - If 1, the gizmo will start an AHB write request once the IP side provides one beat of data of a burst. For IP that can provide quick continuous burst data, this setting provides parallelism between AHB request and IP data transfer. However, for IP that cannot provide quick burst data, this setting will force this gizmo to hold the AHB bus longer, reducing bus efficiency.
   - If 0, gizmo will wait until all beats of data of a burst are provided before starting an AHB write request. With this setting, AHB bus efficiency is realized but IP latency and efficiency may be reduced.
3. RD_DATA: Controls how read data will be returned from the gizmo to the IP side.
   - If 1, all beats of data of a burst have to be available before it indicates to the IP logic that read data is ready. This setting ensures there is no wait-state (bubble) between read data burst, but it will increase latency.
   - If 0, each beat of data of a burst will be sent from gizmo to the IP logic immediately. This setting will reduce latency, but can create non-consecutive data burst or bubble between data.
4. REQ_NEG_CNT: Provides a way to limit (in terms of number of AHB bus clock) how fast/often this master can request the AHB bus. The bigger the number, the slower the rate of AHB request.

(2) AHB Slave Gizmo

1. ENABLE_SPLIT: Controls enabling the split feature of the AHB bus.
   - If 1, the gizmo will always generate split response for a read. If ' DONT_SPLIT_AHB_WR' is set to 0, gizmo also generates split response if it cannot accept write request from the AHB master anymore. This setting can increase AHB bus utilization since it allows other AHB masters to talk to other AHB slaves while this original AHB slave is fetching read data. However, the flip side is that it takes longer time for the original AHB master to get the read data, because the original AHB master has to re-arbitrate for the AHB bus again in order to get to the data it asked for.

- If 0, the gizmo will not generate split response. Instead, it will hold on to the AHB bus until all the requested read data is returned back to the AHB master. This setting will reduce read latency to the master, but decrease AHB bus utilization.

2. FORCE_TO_AHB_SINGLE: Controls how the gizmo treats the AHB master's burst request.
   - If 1, the gizmo will break up the burst request internally into individual single word requests.
   - If 0, the gizmo will burst request internally as burst request.
3. ENB_FAST_REARBITRATE: Controls when the gizmo can allow the original read requested AHB master to re-arbitrate for the AHB bus again so the AHB master can retrieve the originally requested read data.
   - If 1, once first read data of a burst is in the slave gizmo's FIFO, it will allow the original AHB master to re-arbitrate, thus, allowing arbitration to happen in parallel with subsequent read data of a burst. However, if the data burst has wait-states or bubbles, then this setting will decrease AHB bus utilization because the slave gizmo will hold on to the AHB bus longer.
   - If 0, all read data of a burst must be in the slave gizmo's FIFO before it allows the original AHB master to re-arbitrate to retrieve the read data. This increases read data latency, and also increase AHB bus utilization.
4. IP_WR_REQ_IMMEDIATE: Controls when the gizmo will start write data request to the IP logic.
   - If 1, the gizmo will start write request to the IP logic once it gets one write data of a burst from the AHB side. This setting will create non-consecutive/bubbles in a write data burst.
   - If 0, the gizmo will start write request to the IP logic only when it gets all write data of a burst from the AHB side. This setting will create consecutive data burst (no bubbles or wait-states).
5. MAX_IP_BURSTSIZE: Controls the maximum burst size that this gizmo can generate to the IP logic.
6. ACCEPT_AHB_WR_ALWAYS: Controls how the slave gizmo will treat AHB write request.
   - If 1, the gizmo will always accept a write request without checking whether it's FIFOs can accept the write or not. This setting can reduce bus utilization, but can reduce the rate of AHB retry.
   - If 0, the gizmo will check its FIFOs to make sure they have room before accepting the AHB write request. This setting increase bus utilization, but can create a lot of AHB retry.
7. DONT_SPLIT_AHB_WR: Controls whether to split AHB write request when the slave gizmo determines it cannot accept the write request.
   - If 1 and when ENABLE_SPLIT=1, the gizmo will generate split for write if its FIFOs are not ready to accept the AHB write request or data. This setting can improve AHB bus utilization as there are no continuous AHB master retries on the bus.
   - If 0, the gizmo will generate retry response for write if its FIFOs are not ready to accept the AHB write request. Software should leave this bit at 0 since this feature has not been proven.

AHB Gizmo AHB-DMA/Memory Control Register

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00020081 (0b00000000xx0xx010xxxxxxx010xxx001)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
21	0x0	CHANNEL_SPLIT_STALL: If enabled, AHB writes to a second MC channel will stall until all writes to the previous MC channel have completed. This prevents a later write from occurring before an earlier write to a different MC channel. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) – Start AHB write request immediately 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo (AHB-DMA) – Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
8	0x0	WR_WAIT_COMMIT_ON_1K: writes to ahbslv will only be issued to MC when all prior writes in different 1kB pages have reached the point of coherency. This is needed because the MC allows re-ordering of transactions. If this is not set, an entity which polls memory for a descriptor or flag to indicate some other memory has been written, may think something is available in memory when it is not. 0 = DISABLE 1 = ENABLE
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo (memory controller) – Do not split AHB write transaction 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo (memory controller) - Accept AHB write request always. 1= Always accept AHB write request without checking whether there is room in the queue to store the write data. Bypass Memory Controller AHB slave gizmo write queue. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. Memory controller AHB slave gizmos write queue is used in this case. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x0	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	FORCE_TO_AHB_SINGLE: AHB slave gizmo (memory controller) - Force all AHB transaction to single data request transaction 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo (memory controller) - Enable splitting AHB transaction. 0 = DISABLE 1 = ENABLE

21.3.1.2 AHB_GIZMO_APB_DMA_0

AHB Gizmo APB-DMA Control Register

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x000a0000 (0b00000000xxxx1010xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
19	0x1	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = Wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = Transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

21.3.1.3 AHB_MASTER_SWID_0

AHB Master SWID Register

Offset: 0x18 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxx0000000000000xx0000000)

Bit	Reset	Description
21	0x0	MIPIHSI:SWID for MIPIHSI
20	0x0	SDMMC3:SWID for SDMMC3
19	0x0	SDMMC2:SWID for SDMMC2
18	0x0	USB2:SWID for USB2
17	0x0	USB3:SWID for USB3
16	0x0	BSEA:SWID for BSEA
15	0x0	DDS:SWID for DDS
14	0x0	SE:SWID for SE
13	0x0	BSEV:SWID for BSEV
12	0x0	SDMMC4:SWID for SDMMC4
11	0x0	NOR:SWID for NOR
10	0x0	NAND:SWID for NAND
9	0x0	SDMMC1:SWID for SDMMC1
6	0x0	USB1:SWID for USB1
5	0x0	AHBDMA:SWID for AHBDMA
4	0x0	ARC:SWID for ARC
3	0x0	CORESIGHT:SWID for CORESIGHT
2	0x0	VCP:SWID for VCP
1	0x0	COP:SWID for COP
0	0x0	CPU:SWID for CPU

21.3.1.4 AHB_GIZMO_USB_0

AHB Gizmo USB Control Register

Offset: 0x20 │ Read/Write: R/W │ Reset: 0x00020087 (0b00000000xxxxx010xxxxxxxx10xxx111)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

21.3.1.5 AHB_GIZMO_AHB_XBAR_BRIDGE_0

AHB Gizmo AHB XBAR Bridge Control Register

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x0000008d (0bxxxxxxxxxxxxxxxxxxxxxxxx10001101)

Bit	Reset	Description
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
5:4	0x0	MAX_IP_BURSTSIZE: AHB slave gizmo - Maximum allowed IP burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
3	0x1	IMMEDIATE: AHB slave gizmo - Start write request to device immediately. 1 = Start write request on the device side as soon as the AHB master puts data into the gizmos queue. 0 = Start the device write request only when the AHB master has placed all write data into the gizmos queue. 0 = DISABLE 1 = ENABLE
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x0	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA

Bit	Reset	Description
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

21.3.1.6 AHB_GIZMO_CPU_AHB_BRIDGE_0

AHB Gizmo CPU AHB Bridge Control Register

Offset: 0x28 │ Read/Write: R/W │ Reset: 0x00060000 (0b00000000xxxx0110xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master
19	0x0	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = Wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = Transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	0x1	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

21.3.1.7 AHB_GIZMO_COP_AHB_BRIDGE_0

AHB Gizmo COP AHB Bridge Control Register

Offset: 0x2c │ Read/Write: R/W │ Reset: 0x00060000 (0b00000000xxxx0110xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master

Bit	Reset	Description
19	0x0	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = Wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = Transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	0x1	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

21.3.1.8 AHB_GIZMO_XBAR_APB_CTLR_0

AHB Gizmo XBAR APB Control Register

Offset: 0x30 │ Read/Write: R/W │ Reset: 0x00000008 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx001xxx)

Bit	Reset	Description
5:4	0x0	MAX_IP_BURSTSIZE: AHB slave gizmo - Maximum allowed IP burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
3	0x1	IMMEDIATE: AHB slave gizmo - Start write request to device immediately. 1 = Start write request on the device side as soon as the AHB master puts data into the gizmos queue. 0 = Start the device write request only when the AHB master has placed all write data into the gizmos queue. 0 = DISABLE 1 = ENABLE

21.3.1.9 AHB_GIZMO_VCP_AHB_BRIDGE_0

AHB Gizmo VCP AHB Bridge Control Register

Offset: 0x34 | Read/Write: R/W | Reset: 0x00060000 (0b00000000xxxx0110xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
19	0x0	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = Wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = Transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	0x1	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

21.3.1.10 AHB_GIZMO_NAND_0

AHB Gizmo NAND Control Register

Offset: 0x40 | Read/Write: R/W | Reset: 0x000a0000 (0b00000000xxxx1010xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
19	0x1	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = Wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = Transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT

Bit	Reset	Description
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

21.3.1.11 AHB_GIZMO_SDMMC4_0

This is SDMMC4 master gizmo configuration.

AHB Gizmo SDMMC4 Control Register

Offset: 0x48 │ Read/Write: R/W │ Reset: 0x00020087 (0b00000000xxxxx010xxxxxxxx10xxx111)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the

Bit	Reset	Description
		queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

21.3.1.12 AHB_GIZMO_SE_0

AHB Gizmo SE Control Register

Offset: 0x50 │ Read/Write: R/W │ Reset: 0x00020000 (0b00000000xxxx0010xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
19	NO_WAIT	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = Wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP 0 = Transfer each read data from the AHB to the IP immediately 0 = NO_WAIT 1 = WAIT
18	DISABLE	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	DMA_BURST_8WORDS	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16 0 = DMA_BURST_1WORDS

Bit	Reset	Description
		1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

21.3.1.13 AHB_GIZMO_TZRAM_0

AHB Gizmo AHB TZRAM Control Register

Offset: 0x54 │ Read/Write: R/W │ Reset: 0x00000081 (0bxxxxxxxxxxxxxxxxxxxxxxxx10xxx001)

Bit	Reset	Description
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x0	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x0	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

21.3.1.14 AHB_GIZMO_BSEV_0

This is the VDE's BSEV master gizmo configuration.

AHB Gizmo BSE Control Register

Offset: 0x64 │ Read/Write: R/W │ Reset: 0x00020000 (0b00000000xxx00010xxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.

Bit	Reset	Description
20	0x0	RESERVED_GIZMO_BSEV: Reserved for future use
19	0x0	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = Wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = Transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	0x0	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. NOTE: THIS SHOULD NEVER BE SET TO ENABLE (BSEV requires this bit to be 0). 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

21.3.1.15 AHB_GIZMO_BSEA_0

VDE's BSEA master gizmo configuration

AHB Gizmo SCE Control Register

Offset: 0x74 │ Read/Write: R/W │ Reset: 0x00020000 (0b00000000xxxxx010xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. NOTE: THIS SHOULD NEVER BE SET TO ENABLE (BSEV requires this bit to be 0) 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

21.3.1.16 AHB_GIZMO_NOR_0

AHB Gizmo AHB NOR Flash Control Register

Offset: 0x78 │ Read/Write: R/W │ Reset: 0x00000081 (0bxxxxxxxxxxxxxxxxxxxxxxxxx10xxx001)

Bit	Reset	Description
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x0	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x0	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

21.3.1.17 AHB_GIZMO_USB2_0

This is now USB2 master gizmo configuration.

AHB Gizmo USB2 Control Register

Offset: 0x7c │ Read/Write: R/W │ Reset: 0x00020087 (0b00000000xxxxx010xxxxxxxx10xxx111)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

21.3.1.18 AHB_GIZMO_USB3_0

USB3 master gizmo configuration.

AHB Gizmo USB3 Control Register

Offset: 0x80 | Read/Write: R/W | Reset: 0x00020087 (0b00000000xxxxx010xxxxxxxx10xxx111)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA

Bit	Reset	Description
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

21.3.1.19 AHB_GIZMO_SDMMC1_0

SDMMC1 master gizmo configuration.

AHB Gizmo SDMMC1 Control Register

Offset: 0x84 | Read/Write: R/W | Reset: 0x00020087 (0b00000000xxxxx010xxxxxxxx10xxx111)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always.

Bit	Reset	Description
		0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

21.3.1.20 AHB_GIZMO_SDMMC2_0

SDMMC2 master gizmo configuration.

AHB Gizmo SDMMC2 Control Register

Offset: 0x88 │ Read/Write: R/W │ Reset: 0x00020087 (0b00000000xxxxx010xxxxxxxx10xxx111)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

21.3.1.21 AHB_GIZMO_SDMMC3_0

SDMMC3 master gizmo configuration.

AHB Gizmo SDMMC3 Control Register

Offset: 0x8c │ Read/Write: R/W │ Reset: 0x00020087 (0b00000000xxxxx010xxxxxxxx10xxx111)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
18	0x0	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device puts data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Do not split AHB write transaction. 1 = Do not split AHB write transaction ever. 0 (and enable_split=1) = Allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE
6	0x0	ACCEPT_AHB_WR_ALWAYS: AHB slave gizmo - Accept AHB write request always. 1 = Always accept AHB write request without checking whether there is room in the queue to store the write data. 0 = Accept AHB write request only when there is enough room in the queue to store all the write data. 0 = ACCEPT_ON_CHECK 1 = ACCEPT_ON_NOCHECK

Bit	Reset	Description
2	0x1	ENB_FAST_REARBITRATE: AHB slave gizmo - Enable fast re-arbitration. 1 = Allow AHB master re-arbitration as soon as the device returns one read data into the gizmos queue. 0 = Allow AHB master re-arbitration only when the device returns all read data into the gizmos queue. 0 = DISABLE 1 = ENABLE
1	0x1	FORCE_TO_AHB_SINGLE: AHB slave gizmo - Force all AHB transaction to single data request transaction. 1 = Force to single data transaction always. 0 = Do not force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 0 = DISABLE 1 = ENABLE

21.3.1.22 AHB_GIZMO_DDS_0

AHB Gizmo DDS Control Register

Offset: 0x90 │ Read/Write: R/W │ Reset: 0x00020000 (0b00000000xxxx0010xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
19	NO_WAIT	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = Wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP 0 = Transfer each read data from the AHB to the IP immediately 0 = NO_WAIT 1 = WAIT
18	DISABLE	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	DMA_BURST_8WORDS	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer. 01 = burst-of-4. 10 = burst-of-8. 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

21.3.1.23 AHB_GIZMO_MIPIHSI_0

AHB Gizmo MIPIHSI Control Register

Offset: 0x94 | Read/Write: R/W | Reset: 0x00020000 (0b00000000xxxx0010xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
19	NO_WAIT	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = Wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = Transfer each read data from the AHB to the IP immediately 0 = NO_WAIT 1 = WAIT
18	DISABLE	IMMEDIATE: AHB master gizmo (AHB-DMA) - Start AHB write request immediately. 1 = Start the AHB write request immediately as soon as the device has put one write data in the AHB gizmos queue. 0 = Start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	DMA_BURST_8WORDS	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

21.3.1.24 AHB_GIZMO_ARC_0

MC master gizmo configuration.

AHB Gizmo MC Control Register

Offset: 0x98 | Read/Write: R/W | Reset: 0x00060000 (0b00000000xxxx0110xxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter used to indicate the minimum number of clock counts between requests from this AHB master.
19	0x0	RD_DATA: AHB master gizmo - Pack all AHB read data. 1 = Wait for all requested read data to be in the AHB gizmos queue before returning the data back to the IP. 0 = Transfer each read data from the AHB to the IP immediately. 0 = NO_WAIT 1 = WAIT
18	0x1	IMMEDIATE: AHB master gizmo - Start AHB write request immediately. 1 = start the AHB write request immediately as soon as the device puts data in the

Bit	Reset	Description
		AHB gizmos queue. 0 = start the AHB write request only when all the write data has transferred from the device to the AHB gizmos queue. 0 = DISABLE 1 = ENABLE
17:16	0x2	MAX_AHB_BURSTSIZE: AHB master gizmo - Maximum allowed AHB burst size. 00 = single transfer 01 = burst-of-4 10 = burst-of-8 11 = burst-of-16. 0 = DMA_BURST_1WORDS 1 = DMA_BURST_4WORDS 2 = DMA_BURST_8WORDS 3 = DMA_BURST_16WORDS

21.3.1.25 AHB_AHB_WRQ_EMPTY_0

Offset: 0xc4 │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	COP_AHB_WRQ_EMPTY
0	X	CPU_AHB_WRQ_EMPTY

21.4 AHB Memory Controller Slave

The AHB memory controller slave is the path used by AHB bus masters to access the Memory Controller. It provides access to DRAM from the AHB bus and can pre-fetch data.

21.4.1 AHB Memory Pre-Fetcher

The AHB memory controller slave contains a pre-fetcher block. This is intended to improve performance for AHB masters performing sequential reads from DRAM. Performance can be a problem because the AHB bus protocol only allows a single outstanding read request from a master, so the round trip latency limits bandwidth. As there are two level of arbitration, firstly on AHB and secondly within the memory controller, this latency can become large on a busy system.

There are eight such pre-fetchers, allowing this function to be active for up to eight AHB masters.

When the pre-fetcher is enabled, the first read request initiates a speculative read of up to 128 bytes, and subsequent linear reads will return the data directly to the AHB master without going through the memory controller to DRAM.

21.4.1.1 Pre-Fetch Invalidate

The pre-fetch buffer's contents are invalidated under any of the following conditions, in all cases a read refers to a DRAM read by the assigned HB master:

- The programmed time-out value is reached since the last read
- A read occurs which is not sequential with the previous read
- A read occurs which is not the same size as the previous read, unless the corresponding DISABLE_CHECK_SIZE_MASTERx bit is set.
- A read occurs which is past the end of the pre-fetch buffer (this will trigger a new fill of the buffer).
- The pre-fetcher is disabled. A momentary disable and then re-enable can be used to invalidate the buffer.

21.4.1.2 Pre-Fetch Buffer Coherency

As the pre-fetch buffer contains a copy of data in DRAM, there is an inherent coherency risk if the DRAM data is updated. The hardware invalidation mechanisms provide some protection here, but there remains risk of subtle problems arising from this hardware. If an AHB master shows errors that appear to arise from stale data then a reasonable experiment is to disable the pre-fetcher to see if that cures the problem. If so, the following guidelines may help.

Problems have been seen for control structures such as USB Transfer Descriptors.

Two approaches can resolve this problem:

1. Pad the data

As the pre-fetcher invalidates the buffer for any non-sequential read, placing some padding will prevent pre-fetch issues. A problem can be triggered by using something like this:

```
struct dtd { unsigned HW_descriptor[K] ; } ; struct dtd dtd_array[N] ;
```

This makes the hardware descriptors structures as read by the USB DMA sequential, and so pre-fetchable.
If the definition is modified as:

```
struct dtd { unsigned HW_descriptor[K] ; unsigned Padding ; } ; struct dtd dtd_array[N] ;
```

Then reads by USB DMA of consecutive hardware descriptors become **not** sequential and so there is no coherency issue as the pre-fetch buffer will be invalidated.

2. Invalidate the pre-fetch buffer

After updating a memory structure, the pre-fetch buffer can be invalidated by disabling the pre-fetcher and then immediately re-enabling it.

21.4.2 AHB Memory Controller Slave Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

21.4.2.1 AHB_AHB_MEM_PREFETCH_CFG5_0

0x6000_C0CC: ahbslv prefetch cfg5 --> reset value = 0x1883.0800.

Offset: 0xcc │ Read/Write: R/W │ Reset: 0x14800800 (0b00010100100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable
30:26	0x5	AHB_MST_ID: 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3

Bit	Reset	Description
		18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 16 will use n=16.
15:0	0x800	INACTIVITY_TIMEOUT

21.4.2.2 AHB_AHB_MEM_PREFETCH_CFG6_0

0x6000_C0D0: ahbslv prefetch cfg6 --> reset value = 0x1883.0800.

Offset: 0xd0 │ Read/Write: R/W │ Reset: 0x18800800 (0b00011000100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable
30:26	0x6	AHB_MST_ID: USB 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 16 will use n=16.
15:0	0x800	INACTIVITY_TIMEOUT

21.4.2.3 AHB_AHB_MEM_PREFETCH_CFG7_0

0x6000_C0D4: ahbslv prefetch cfg7 --> reset value = 0x1883.0800.

Offset: 0xd4 │ Read/Write: R/W │ Reset: 0x1c800800 (0b00011100100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable
30:26	0x7	AHB_MST_ID: USB 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. any value greater than 16 will use n=16.
15:0	0x800	INACTIVITY_TIMEOUT

21.4.2.4 AHB_AHB_MEM_PREFETCH_CFG8_0

0x6000_C0D8: ahbslv prefetch cfg8 --> reset value = 0x1883.0800.

Offset: 0xd8 │ Read/Write: R/W │ Reset: 0x20800800 (0b00100000100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable
30:26	0x8	AHB_MST_ID: USB 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08

Bit	Reset	Description
		9 = SDIO1 10 = NAND_FLASH 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 16 will use n=16.
15:0	0x800	INACTIVITY_TIMEOUT

21.4.2.5 AHB_AHB_MEM_PREFETCH_CFG_X_0

If DISABLE_CHECK_SIZE is 0, then only read requests that have the exact same size as the original read request that kick-started the prefetch process will cause a "hit". In addition, the address must be the exact next one in the sequence.

For instance, if the first request on a prefetch-enabled AHB Master arrives with SIZE=ONE_BYTE and ADDR[31:0]=0x3, then the next access must have SIZE=ONE_BYTE and ADDR[31:0]=0x4 in order to be considered a hit.

If DISABLE_CHECK_SIZE is 1, then a read request will hit as long as the incoming ADDR[31:4]matches the expected_ADDR[31:4]. expected_ADDR[31:4] is always either "last ADDR[31:4]" or "last ADDR[31:4] + 1", where last ADDR is the prior read request actually issued by the AHB Master(as opposed to the last request to the CIF, which could have been a speculative read).

expected_ADDR[31:4] is always "last ADDR[31:4]" unless "SIZE" field in the last request uses the last byte in the 16-byte CIF word.

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

Bit	Reset	Description
15	0x0	DISABLE_ADDR_BNDY_CHK_MST8:
14	0x0	DISABLE_ADDR_BNDY_CHK_MST7:
13	0x0	DISABLE_ADDR_BNDY_CHK_MST6:
12	0x0	DISABLE_ADDR_BNDY_CHK_MST5:
11	0x0	DISABLE_ADDR_BNDY_CHK_MST4:
10	0x0	DISABLE_ADDR_BNDY_CHK_MST3:
9	0x0	DISABLE_ADDR_BNDY_CHK_MST2:
8	0x0	DISABLE_ADDR_BNDY_CHK_MST1
7	0x0	DISABLE_CHECK_SIZE_MASTER8:

Bit	Reset	Description
6	0x0	DISABLE_CHECK_SIZE_MASTER7:
5	0x0	DISABLE_CHECK_SIZE_MASTER6:
4	0x0	DISABLE_CHECK_SIZE_MASTER5:
3	0x0	DISABLE_CHECK_SIZE_MASTER4:
2	0x0	DISABLE_CHECK_SIZE_MASTER3:
1	0x0	DISABLE_CHECK_SIZE_MASTER2:
0	0x0	DISABLE_CHECK_SIZE_MASTER1:

21.4.2.6 AHB_ARBITRATION_XBAR_CTRL_0

XBAR Control Register

Offset: 0xe0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00xxxxxxxxxxxxx000)

Bit	Reset	Description
17	0x0	SMMU_INIT_DONE: Software should set this bit when SMMU has been initialized 0 = NOT_DONE 1 = DONE
16	0x0	MEM_INIT_DONE: Software should set this bit when memory has been initialized 0 = NOT_DONE 1 = DONE
2	0x0	PPSB_STOPCLK_ENABLE: Deprecated. 0 = DISABLE 1 = ENABLE
1	0x0	HOLD_DIS: By default CPU accesses to IRAMs will be held if there are any pending requests from the AHB to the IRAMs. This is done to avoid data coherency issues. If software handles coherency then this can be turned off to improve performance. Software writes to modify. 0 = ENABLE 1 = DISABLE
0	0x0	POST_DIS: Software writes to modify 0 = ENABLE 1 = DISABLE

21.4.2.7 AHB_AHB_MEM_PREFETCH_CFG3_0

See the description of the pre-fetcher above. 6000:C0E4: ahbslv prefetch cfg3 --> reset value = 0x1483.0800.

Offset: 0xe4 │ Read/Write: R/W │ Reset: 0x0c800800 (0b00001100100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable. 0=disable
30:26	0x3	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH

Bit	Reset	Description
		11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNISED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 26 will use n=26.
15:0	0x800	INACTIVITY_TIMEOUT: 2048 cycles

21.4.2.8 AHB_AHB_MEM_PREFETCH_CFG4_0

See the description of the pre-fetcher above. 6000:C0E8: ahbslv prefetch cfg4 --> reset value = 0x1483.0800.

Offset: 0xe8 │ Read/Write: R/W │ Reset: 0x10800800 (0b00010000100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable. 0=disable
30:26	0x4	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 26 will use n=26.

Bit	Reset	Description
15:0	0x800	INACTIVITY_TIMEOUT: 2048 cycles

21.4.2.9 AHB_AVP_PPCS_RD_COH_STATUS_0

0x6000_C0EC: ARM7 outstanding rd/wr

Offset: 0xec │ Read/Write: RO │ Reset: 0x000X000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
16	X	RDS_OUTSTANDING
0	X	WRS_OUTSTANDING

NV_ahbslvmem prefetch

The NV_ahbslvmem prefetch feature is intended to reduce latency and improve overall bandwidth for AHB Masters doing reads to SDRAM. AHB only allows one outstanding read transaction at a time. When enabled, and kick-started by a first read request (which will "miss" since there would be nothing in the prefetch FIFO) the prefetcher makes speculative read requests to the memory controller in consecutive progression of linear address. Without this feature, an AHB Master will suffer roundtrip latency through the gizmo, PPCS, CIF, MC arbitration, and all the read data passing for every AHB transaction.

Address Boundaries for prefetching will stop the sequential prefetch process from making additional speculations. This is useful to help prevent coherency issues -- software needs ways to stop the prefetcher from prefetching data before the data has been written in memory. If an AHB master can be known to make 256 byte transfers that are aligned accesses, ADDR_BNDRY should be set to a value of 4.

INACTIVITY_TIMEOUT is intended to prevent coherency problems. If no read request from the prefetch-enabled master is observed after the number of cycles specified in the inactivity timeout counter, then any speculatively prefetched read data is invalidated.

There are eight AHB masters that can be enabled for prefetching. Each prefetch buffer can hold up to 8 entries (of 128 bits each entry) of prefetched data.

21.4.2.10 AHB_AHB_MEM_PREFETCH_CFG1_0

0x6000_C0F0: ahbslv prefetch cfg1 --> reset value = 0x1483.0800.

Offset: 0xf0 │ Read/Write: R/W │ Reset: 0x04800800 (0b00000100100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable
30:26	0x1	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 26 will use n=26.
15:0	0x800	INACTIVITY_TIMEOUT: 2048 cycles

21.4.2.11 AHB_AHB_MEM_PREFETCH_CFG2_0

See the description of the pre-fetcher above. 0x6000_C0F4: ahbslv prefetch cfg2 --> reset value = 0x1883.0800.

Offset: 0xf4 │ Read/Write: R/W │ Reset: 0x08800800 (0b00001000100xxxxx0000100000000000)

Bit	Reset	Description
31	0x0	ENABLE: 1=enable, 0=disable
30:26	0x2	AHB_MST_ID: AHBDMA master 0 = CPU 1 = COP 2 = VCP 3 = CSITE 4 = IDE 5 = AHBDMA 6 = USB 7 = APBDMA 8 = UNUSED_08 9 = SDIO1 10 = NAND_FLASH 11 = SNOR 12 = HSMMC 13 = BSEV 14 = SE 15 = UNUSED_0F 16 = BSEA 17 = USB3 18 = USB2 19 = SDIO2 20 = UNUSED_14 21 = UNUSED_15 22 = UNUSED_16 23 = UNUSED_17 24 = UNUSED_18 25 = UNUSED_19 26 = UNUSED_1A 27 = UNUSED_1B 28 = UNUSED_1C 29 = UNUSED_1D 30 = UNUSED_1E 31 = UNUSED_1F
25:21	0x4	ADDR_BNDRY: 2^(n+6) byte boundary. Any value greater than 26 will use n=26.
15:0	0x800	INACTIVITY_TIMEOUT

21.4.2.12 AHB_AHBSLVMEM_STATUS_0

0x6000_C0F8: ahbslv outstanding rd, rdque_empty status

Offset: 0xf8 │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	PPCS_RDS_OUTSTANDING
0	X	GIZMO_IP_RDQUE_EMPTY

21.4.2.13 AHB_ARBITRATION_AHB_MEM_WRQUE_MST_ID_0

0x6000_C0FC: AHB Memory Write Queue AHB Master ID Register

Offset: 0xfc | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000000000000000000000)

Bit	Reset	Description
30:0	0x0	AHB_MASTER_ID: 0 = There is no write data in the write queue from that AHB master.

21.4.2.14 AHB_ARBITRATION_CPU_ABORT_ADDR_0

0x6000_C100: CPU Abort Address Register

Offset: 0x100 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ADDR: Instruction Address which caused the abort.

21.4.2.15 AHB_ARBITRATION_CPU_ABORT_INFO_0

0x6000_C104: CPU Abort Info Register

Offset: 0x104 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	IRAMA: Abort occurred due to an iRAMa protection violation 0 = ABT_DIS 1 = ABT_EN
14	X	IRAMB: Abort occurred due to an iRAMb protection violation 0 = ABT_DIS 1 = ABT_EN
13	X	IRAMC: Abort occurred due to an iRAMc protection violation 0 = ABT_DIS 1 = ABT_EN
12	X	IRAMD: Abort occurred due to an iRAMd protection violation 0 = ABT_DIS 1 = ABT_EN
11	X	INV_IRAM: Abort occurred due to an access to invalid iRAM address space 0 = ABT_DIS 1 = ABT_EN
10	X	PPSB: Abort occurred due to a PPSB protection violation 0 = ABT_DIS 1 = ABT_EN
9	X	APB: Abort occurred due to an APB protection violation 0 = ABT_DIS 1 = ABT_EN
8	X	AHB: Abort occurred due to an AHB protection violation 0 = ABT_DIS 1 = ABT_EN

Bit	Reset	Description
7	X	CACHE: Abort occurred due to a Cache protection violation 0 = ABT_DIS 1 = ABT_EN
6	X	PROTECTION: TRUE for any protection violation 0 = ABT_DIS 1 = ABT_EN
5	X	ALIGN: TRUE for abort caused by Misalignment (i.e., word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
4	X	BADSIZE: TRUE for abort caused by Bad Size (i.e., word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
3	X	WRITE: Aborted transaction was a Write 0 = ABT_DIS 1 = ABT_EN
2	X	DATA: Aborted transaction was a Data access 0 = ABT_DIS 1 = ABT_EN
1:0	X	SIZE: Aborted transaction Request Size 00=byte 01=halfword 10=word 0 = BYTE_ABT 1 = HWORD_ABT 2 = WORD_ABT

21.4.2.16 AHB_ARBITRATION_COP_ABORT_ADDR_0

0x6000_C108: CPU Abort Address Register

Offset: 0x108 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	ADDR: Instruction Address which caused the abort

21.4.2.17 AHB_ARBITRATION_COP_ABORT_INFO_0

0x6000_C10C: COP Abort Info Register

Offset: 0x10c │ Read/Write: RO │ Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	IRAMA: Abort occurred due to an iRAMa protection violation 0 = ABT_DIS 1 = ABT_EN
14	X	IRAMB: Abort occurred due to an iRAMb protection violation 0 = ABT_DIS 1 = ABT_EN
13	X	IRAMC: Abort occurred due to an iRAMc protection violation 0 = ABT_DIS 1 = ABT_EN
10	X	PPSB: Abort occurred due to a PPSB protection violation 0 = ABT_DIS 1 = ABT_EN
9	X	APB: Abort occurred due to an APB protection violation 0 = ABT_DIS 1 = ABT_EN
8	X	AHB: Abort occurred due to an AHB protection violation 0 = ABT_DIS 1 = ABT_EN
7	X	CACHE: Abort occurred due to a Cache protection violation 0 = ABT_DIS 1 = ABT_EN
6	X	PROTECTION: TRUE for any protection violation 0 = ABT_DIS 1 = ABT_EN
5	X	ALIGN: TRUE for abort caused by Misalignment (i.e., word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
4	X	BADSIZE: TRUE for abort caused by Bad Size (i.e., word access at odd byte address) 0 = ABT_DIS 1 = ABT_EN
3	X	WRITE: Aborted transaction was a Write 0 = ABT_DIS 1 = ABT_EN
2	X	DATA: Aborted transaction was a Data access 0 = ABT_DIS 1 = ABT_EN

Bit	Reset	Description
1:0	X	SIZE: Aborted transaction Request Size 00=byte 01=halfword 10=word 0 = BYTE_ABT 1 = HWORD_ABT 2 = WORD_ABT

21.4.2.18 AHB_AHB_SPARE_REG_0

0x6000_C110: AHB Spare Register Bits

Offset: 0x110 │ Read/Write: R/W │ Reset: 0x00000060 (0b00000000000000000000000001100000)

Bit	Reset	Description
31:5	0x0	AHB_SPARE_REG: AHB Spare Register
4:0	0x0	CSITE_PADMACRO_TRIM_SEL: Trimmer select register for CoreSight clock pad macro.

21.4.2.19 AHB_XBAR_SPARE_REG_0

0x6000_C114: XBAR Spare Register Bits

Offset: 0x114 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:6	0x0	UNUSED: XBAR Diagnostic Register Spare Bits
5	0x0	MASK_PEND_IRAM_REQ: XBAR Diagnostic Register to mask pending IRAM request while arbitration
4	0x0	KILL_NEXT_REQ_ON_ABORT_UCQ: XBAR Diagnostic Register to kill next request on ABORT from UCQ
3	0x0	KILL_NEXT_REQ_ON_ABORT_AHB: XBAR Diagnostic Register to kill next request on ABORT from AHB Bridge
2	0x0	KILL_NEXT_REQ_ON_ABORT_VCP2: XBAR Diagnostic Register to kill next request on ABORT from VCP2
1	0x0	KILL_NEXT_REQ_ON_ABORT_COP: XBAR Diagnostic Register to kill next request on ABORT from COP
0	0x0	KILL_NEXT_REQ_ON_ABORT_APC: XBAR Diagnostic Register to kill next request on ABORT from APC

0x6000_C110 - 0x6001_C23C;

0x130 = 304 decimal = 76 registers * 4 bytes each

reserve [76] incr4;

0x6000_C240 - 0x6001_C28C:

NV_AVPC_MCCIF Configuration

New base AVPC_MCCIF 0x6001_C040;

0x6000_C240: rdcl_rdfast, wrmc_clle2x, rdmc_rdfast, wrcl_mcle2x (client or memory side clock control)

0x6000_C244: avpc(arm7)w_wcoal_tmval (write coalesce timeout value)

0x6000_C248-84: 16 reserved registers

0x6000_C288-8C: reserved unused (for nice alignment)

21.4.2.20 AHB_EMUCIF_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register

Note: The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility.

The clock enable fields of this register control the second-level clock gating for the MC side of the MCCIF. A '1' written to the override field will result in that clock reverting to the legacy mode of operation (where the msc*clk is on whenever the client clock is enabled.

Offset: 0x11c │ Byte Offset: 0x470 │ Read/Write: R/W │ Reset: 0x00010000 (0bxxxxxxxxxxxxxxx01xxxxxxxxxxxxxxx)

Bit	Reset	Description
17	0x0	SYS_REGS_EMUCIF_RCLK_OVERRIDE
16	0x1	SYS_REGS_EMUCIF_WCLK_OVERRIDE

21.4.2.21 AHB_AVPC_MCCIF_FIFOCTRL_0

Note: The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility.

The clock enable fields of this register control the second-level clock gating for the MC side of the MCCIF.

A '1' written to the override field will result in that clock reverting to the legacy mode of operation (where the msc*clk is on whenever the client clock is enabled.

Memory Client Interface FIFO Control Register

Offset: 0x120 │ Byte Offset: 0x480 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00xxxxxxxxxxx0000)

Bit	Reset	Description
17	0x0	AVPC_RCLK_OVERRIDE
16	0x0	AVPC_WCLK_OVERRIDE
3	DISABLE	AVPC_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	AVPC_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	DISABLE	AVPC_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	AVPC_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

21.4.2.22 AHB_TIMEOUT_WCOAL_AVPC_0

Note: Write coalescing is no longer supported by the MCCIF clients. Registers are retained for software compatibility but are not used by the hardware.

Write Coalescing Time-Out Register

Offset: 0x124 | Byte Offset: 0x490 | Read/Write: R/W | Reset: 0x00000032 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00110010)

Bit	Reset	Description
7:0	0x32	AVPCARM7W_WCOAL_TMVAL

21.4.2.23 AHB_MPCORELP_MCCIF_FIFOCTRL_0

Note: The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility

The clock enable fields of this register control the second-level clock gating for the MC side of the MCCIF.

A '1' written to the override field will result in that clock reverting to the legacy mode of operation (where the msc*clk is on whenever the client clock is enabled.

Memory Client Interface FIFO Control Register

Offset: 0x128 | Byte Offset: 0x4a0 | Read/Write: R/W | Reset: 0x00010000 (0bxxxxxxxxxxxxxx01xxxxxxxxxxxxxxxx)

Bit	Reset	Description
17	0x0	SYS_REGS_MPCORELP_RCLK_OVERRIDE
16	0x1	SYS_REGS_MPCORELP_WCLK_OVERRIDE

21.4.2.24 AHB_MPCORE_MCCIF_FIFOCTRL_0

Note: The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility

The clock enable fields of this register control the second-level clock gating for the MC side of the MCCIF.

A '1' written to the override field will result in that clock reverting to the legacy mode of operation (where the msc*clk is on whenever the client clock is enabled.

Memory Client Interface FIFO Control Register

Offset: 0x12c │ Byte Offset: 0x4b0 │ Read/Write: R/W │ Reset: 0x00010000 (0bxxxxxxxxxxxxxx01xxxxxxxxxxxxxxxx)

Bit	Reset	Description
17	0x0	SYS_REGS_MPCORE_RCLK_OVERRIDE
16	0x1	SYS_REGS_MPCORE_WCLK_OVERRIDE

[THIS PAGE INTENTIONALLY LEFT BLANK]

22.0 APB

22.1 APB Miscellaneous Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

This section describes a number of system control registers that are grouped together in the aptly named miscellaneous registers section. These registers are all present on the APB bus.

22.1.1 Strap Registers

This Boot Strapping register gets updated on power-on reset via the Pull-up/pull-down on the Strap I/Os. Based on strapping options set, the chip determines where to boot from. NOR FLASH WIDTH, MIO_WIDTH are also determined by Boot strap bits.

Some of these bits are used only during the design phase.

22.1.1.1 APB_MISC_PP_STRAPPING_OPT_A_0

Strapping Options Register

Offset: 0x8 │ Read/Write: R/W │ Reset: 0xXXX000X0 (0bxxxxxxxxxxxxxxxxxxxxxxxx00xxxxxxx0)

Bit	Reset	Description
29:26	X	BOOT_SELECT: Read at power-on reset time from gmi_ad[3:0] strap pads. 0 = MPCORE_G 1 = MPCORE_LP
25	X	BOOT_SRC_USB_RECOVERY_MODE: Read at power-on reset time from gmi_oe_n strap pad 0 = DISABLED 1 = ENABLED
24	X	BOOT_SRC_NOR_BOOT: Read at power-on reset time from gmi_wr_n strap pad 0 = IROM 1 = NOR
23:22	X	ARM_JTAG: Read at power-on reset time from gmi_ad[19:18] strap pads. 00=Serial_JTAG 01=CPU_only 10=COP_only 11=Serial_JTAG (same as 00 case) 0 = SERIAL 1 = CPU 2 = COP 3 = SERIAL_ALT
9	0x0	BOOT_FAST_UART: UART Boot speed from TMC JTAG config bit 0=57600 baud 1=Oscillator Frequency (1 bit per OSC clock) 0 = SLOW 1 = FAST
8	RSVD1	MIO_WIDTH: 0 = RSVD1 1 = RSVD2
7:4	X	RAM_CODE: Read at power-on reset time from gmi_ad[7:4] strap pads.

Bit	Reset	Description
0	RSVD1	NOR_WIDTH: 0 = RSVD1 1 = RSVD2

22.1.2 JTAG Configuration Register

22.1.2.1 APB_MISC_PP_CONFIG_CTL_0

Configuration Control Register

The use of the SO bits below has been deprecated. Keep these bits disabled at all times.

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x00000040 (0bxxxxxxxxxxxxxxxxxxxxxxxxx01xxxx00)

Bit	Reset	Description
7	DISABLE	TBE: 0 = Disable ; 1 = Enable RTCK Daisy chaining 0 = DISABLE 1 = ENABLE
6	ENABLE	JTAG: 0 = Disable Debug ; 1 = Enable JTAG DBGEN 0 = DISABLE 1 = ENABLE
1	DISABLE	XBAR_SO_DEFAULT: Deprecated -- keep disabled -- default SO bit for non-CPU XBAR clients 0 = DISABLE 1 = ENABLE
0	DISABLE	CPU_XBAR_SO_ENABLE: Deprecated -- keep disabled --Enable CPU SO bit to propagate to XBAR 0 = DISABLE 1 = ENABLE

22.1.2.2 APB_MISC_PP_PINMUX_GLOBAL_0_0

Global Pinmux Control Register

When both CLAMP_INPUTS_WHEN_TRISTATED is set to ENABLE and the TRISTATE field is set to TRISTATE (1b1), the inputs to the core are clamped to zero. Software must set this bit before taking any controller using a pinmuxed pin out of reset.

Offset: 0x40 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	DISABLE	CLAMP_INPUTS_WHEN_TRISTATED: 0 = Do not clamp inputs when tristated; 1 = Clamp inputs when tristated 0 = DISABLE 1 = ENABLE

22.1.3 PULL_UP/PULL_DOWN Control Register

Controls the pull_up/pull_down inputs of the pads. Used to eliminate external components on interfaces that need pullup/pulldown. Those are weak internal pullup/pulldowns

- NORMAL setting means pad is neither pulled up (driving weak 1) or pulled down (driving weak 0)
- PULL_DOWN selection means pad is driving weak 0
- PULL_UP selection means pad is driving weak 1

WARNING! Driving an internal pull-up when the pad has an external pull-down and vice versa will cause a significant increase in power consumption.

Table 53: Cross Reference of Fields with Real Ball Names

Field Name	Controls Balls
XM2D_PU_PD	DDR_DQ0,DDR_DQ1,...,DDR_DQ31
XM2C_PU_PD	MIO_IORDY,DDR_DM0,..,DDR_DM3,DDR_DQS0,...,DDR_DQS3,DDR_A0,..,DDR_A13,DDR_BA0,DDR_BA1,DDR_CS0_,DDR_CS1_,DDR_CKE,DDR_RAS_,DDR_CAS_,DDR_WE_
DDRC_PU_PD	DDR_COMP_PU, DDR_COMP_PD

22.1.3.1 APB_MISC_PP_PULLUPDOWN_REG_C_0

Offset: 0xa8 │ Read/Write: R/W │ Reset: 0x00000000 (0b000000xxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	NORMAL	XM2C_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
29:28	NORMAL	XM2D_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
27:26	NORMAL	DDRC_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

22.1.3.2 APB_MISC_GP_HIDREV_0

Chip ID Revision Register

Offset: 0x804 │ Read/Write: RO │ Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	MINORREV: Chip ID minor revision
15:8	X	CHIPID: Chip ID
7:4	X	MAJORREV: Chip ID major revision (0: Emulation, 1-15: Silicon) 0 = EMULATION 1 = A01
3:0	X	HIDFAM: Chip ID family register. 0 = GPU 1 = HANDHELD 2 = BR_CHIPS 3 = CRUSH 4 = MCP 5 = CK 6 = VAIO 7 = HANDHELD_SOC

22.1.4 Pad Control Registers

The registers below control pad behavior. For each digital pad, the following controls can be used to tune pad performance, functionality, and power consumption. The pads controlled by each pad group register can be found in the Pinmux section of this document.

- **HSM_EN** - High speed mode - active high, enables high speed mode for driver and receiver for better matching of the rise/fall delay in outbound and inbound paths. Use it for clocks and the high speed signaling where the matching timings are of importance.
- **SCHMT_EN** - Schmitt enable - active high, enables the Schmitt Trigger Type of I/P receiver. Default is Inverter Type of receiver.
- **LPMD** - Low power mode - select low power modes (different impedance and current value). The table depicts the function of low power mode

Table 54: Low Power Mode Functions

LPMD1	LPMD0	Power Mode Selected
0	0	X/8 Current Setting. Lowest Power or Current (8*Z ohm)
0	1	X/4 Current Setting (4*Z ohm)
1	0	X/2 Current Setting (2*Z ohm)
1	1	X Current Setting (Z ohm = 50 ohm). Highest Power or Current.

- CAL_DRVDN - drive down (falling edge) - Driver Output Pull-Down drive strength code.
- CAL_DRVUP - drive up (rising edge) - Driver Output Pull-Up drive strength code. Works with combination of LMPD bits. For lower power modes, higher drive strength are masked. See table below:

LPMD1	LPMD0	CAL_DRV*4	CAL_DRV*3	CAL_DRV*2	CAL_DRV*1	CAL_DRV*0
0	0	Masked to 0	Masked to 0	Masked to 0	Pass Code	Pass Code
0	1	Masked to 0	Masked to 0	Pass Code	Pass Code	Pass Code
1	0	Masked to 0	Pass Code	Pass Code	Pass Code	Pass Code
1	1	Pass Code	Pass Code	Pass Code	Pass Code	Pass Code

- DRVDN_SLWR - Driver Output Rising Edge Slew 2-bit control code.
  Code 11 is least slewing of signal, code 00 is highest slewing of the signal.
- DRVUP_SLWF -Driver Output Falling Edge Slew 2-bit control code.
  Code 11 is least slewing of signal, code 00 is highest slewing of the signal.

22.1.4.1 APB_MISC_GP_AOCFG1PADCTRL_0

AOCFG1 Pad Control Register

Offset: 0x868 │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_AOCFG1_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_AOCFG1_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_AOCFG1_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_AOCFG1_CAL_DRVDN
5:4	0x3	CFG2TMC_AOCFG1_LPMD: AOCFG1 data pins low power mode select
3	0x0	CFG2TMC_AOCFG1_SCHMT_EN: AOCFG1 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_AOCFG1_HSM_EN: AOCFG1 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.2 APB_MISC_GP_AOCFG2PADCTRL_0

AOCFG2 Pad Control Register

Offset: 0x86c │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_AOCFG2_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_AOCFG2_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_AOCFG2_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_AOCFG2_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_AOCFG2_LPMD: AOCFG2 data pins low power mode select
3	0x0	CFG2TMC_AOCFG2_SCHMT_EN: AOCFG2 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_AOCFG2_HSM_EN: AOCFG2 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.3 APB_MISC_GP_ATCFG1PADCTRL_0

ATCFG1 Pad Control Register

Offset: 0x870 | Read/Write: R/W | Reset: 0x01318000 (0b0000x0010011x0011000xxxx00xx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_ATCFG1_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_ATCFG1_CAL_DRVDN_SLWR
26:20	0x13	CFG2TMC_ATCFG1_CAL_DRVUP
18:12	0x18	CFG2TMC_ATCFG1_CAL_DRVDN
7:6	0x0	CFG2TMC_ATCFG1_DRV_TYPE
3	0x0	CFG2TMC_ATCFG1_SCHMT_EN: ATCFG1 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG1_HSM_EN: ATCFG1 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.4 APB_MISC_GP_ATCFG2PADCTRL_0

ATCFG2 Pad Control Register

Offset: 0x874 | Read/Write: R/W | Reset: 0x01318000 (0b0000x0010011x0011000xxxx00xx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_ATCFG2_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_ATCFG2_CAL_DRVDN_SLWR
26:20	0x13	CFG2TMC_ATCFG2_CAL_DRVUP
18:12	0x18	CFG2TMC_ATCFG2_CAL_DRVDN
7:6	0x0	CFG2TMC_ATCFG2_DRV_TYPE
3	0x0	CFG2TMC_ATCFG2_SCHMT_EN: ATCFG2 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG2_HSM_EN: ATCFG2 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.5 APB_MISC_GP_ATCFG3PADCTRL_0

ATCFG3 Pad Control Register

Offset: 0x878 | Read/Write: R/W | Reset: 0x01318000 (0b0000x0010011x0011000xxxx00xx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_ATCFG3_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_ATCFG3_CAL_DRVDN_SLWR

Bit	Reset	Description
26:20	0x13	CFG2TMC_ATCFG3_CAL_DRVUP
18:12	0x18	CFG2TMC_ATCFG3_CAL_DRVDN
7:6	0x0	CFG2TMC_ATCFG3_DRV_TYPE
3	0x0	CFG2TMC_ATCFG3_SCHMT_EN: ATCFG3 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG3_HSM_EN: ATCFG3 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.6 APB_MISC_GP_ATCFG4PADCTRL_0

ATCFG4 Pad Control Register

Offset: 0x87c │ Read/Write: R/W │ Reset: 0x01318000 (0b0000x0010011x0011000xxxx00xx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_ATCFG4_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_ATCFG4_CAL_DRVDN_SLWR
26:20	0x13	CFG2TMC_ATCFG4_CAL_DRVUP
18:12	0x18	CFG2TMC_ATCFG4_CAL_DRVDN
7:6	0x3	CFG2TMC_ATCFG4_DRV_TYPE
3	0x0	CFG2TMC_ATCFG4_SCHMT_EN: ATCFG4 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG4_HSM_EN: ATCFG4 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.7 APB_MISC_GP_ATCFG5PADCTRL_0

ATCFG5 Pad Control Register

Offset: 0x880 │ Read/Write: R/W │ Reset: 0xf0b48030 (0b1111xxxx1011010010xxxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_ATCFG5_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_ATCFG5_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_ATCFG5_CAL_DRVUP
18:14	0x12	CFG2TMC_ATCFG5_CAL_DRVDN
5:4	0x3	CFG2TMC_ATCFG5_LPMD: ATCFG5 data pins low power mode select
3	0x0	CFG2TMC_ATCFG5_SCHMT_EN: ATCFG5 data pins Schmitt enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	CFG2TMC_ATCFG5_HSM_EN: ATCFG5 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.8 APB_MISC_GP_CDEV1CFGPADCTRL_0

CDEV1CFG Pad Control Register

Offset: 0x884 │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_CDEV1CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_CDEV1CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_CDEV1CFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_CDEV1CFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_CDEV1CFG_LPMD: CDEV1CFG data pins low power mode select
3	0x0	CFG2TMC_CDEV1CFG_SCHMT_EN: CDEV1CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CDEV1CFG_HSM_EN: CDEV1CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.9 APB_MISC_GP_CDEV2CFGPADCTRL_0

CDEV2CFG Pad Control Register

Offset: 0x888 │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_CDEV2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_CDEV2CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_CDEV2CFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_CDEV2CFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_CDEV2CFG_LPMD: CDEV2CFG data pins low power mode select

Bit	Reset	Description
3	0x0	CFG2TMC_CDEV2CFG_SCHMT_EN: CDEV2CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CDEV2CFG_HSM_EN: CDEV2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.10 APB_MISC_GP_DAP1CFGPADCTRL_0

DAP1CFG Pad Control Register

Offset: 0x890 │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP1CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP1CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_DAP1CFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_DAP1CFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_DAP1CFG_LPMD: DAP1CFG data pins low power mode select
3	0x0	CFG2TMC_DAP1CFG_SCHMT_EN: DAP1CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP1CFG_HSM_EN: DAP1CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.11 APB_MISC_GP_DAP2CFGPADCTRL_0

DAP2CFG Pad Control Register

Offset: 0x894 │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP2CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_DAP2CFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS

Bit	Reset	Description
16:12	0x9	CFG2TMC_DAP2CFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_DAP2CFG_LPMD: DAP2CFG data pins low power mode select
3	0x0	CFG2TMC_DAP2CFG_SCHMT_EN: DAP2CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP2CFG_HSM_EN: DAP2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.12 APB_MISC_GP_DAP3CFGPADCTRL_0

DAP3CFG Pad Control Register

Offset: 0x898 │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP3CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP3CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_DAP3CFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_DAP3CFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_DAP3CFG_LPMD: DAP3CFG data pins low power mode select
3	0x0	CFG2TMC_DAP3CFG_SCHMT_EN: DAP3CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP3CFG_HSM_EN: DAP3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.13 APB_MISC_GP_DAP4CFGPADCTRL_0

DAP4CFG Pad Control Register

Offset: 0x89c │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP4CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP4CFG_CAL_DRVDN_SLWR

Bit	Reset	Description
24:20	0xe	CFG2TMC_DAP4CFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_DAP4CFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_DAP4CFG_LPMD: DAP4CFG data pins low power mode select
3	0x0	CFG2TMC_DAP4CFG_SCHMT_EN: DAP4CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP4CFG_HSM_EN: DAP4CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.14 APB_MISC_GP_DBGCFGPADCTRL_0

DBGCFG Pad Control Register

Offset: 0x8a0 │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DBGCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DBGCFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_DBGCFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_DBGCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DBGCFG_LPMD: DBGCFG data pins low power mode select
3	0x0	CFG2TMC_DBGCFG_SCHMT_EN: DBGCFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DBGCFG_HSM_EN: DBGCFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.15 APB_MISC_GP_SDIO3CFGPADCTRL_0

Controls for BDSDMEM pads of SDMMC3

The following calibration codes need to be used for the pad under default condition. In general, the calibration pad will provide the code for the pad. To bypass calibration and provide the default code for the required impedance, these values (in decimal) should be used.

Table 55: DRV Codes

Supply	33 ohms		50 ohms		66 ohms		100 ohms	
	Up	DN	Up	DN	Up	DN	Up	DN
1.8V	70	54	42	32	34	23	19	
3.3V			36	20				

SDIO3CFG Pad Control Register

Offset: 0x8b0 | Read/Write: R/W | Reset: 0x01318000 (0b0000x0010011x0011000xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SDIO3CFG_CAL_DRVUP_SLWF. Maps to MSB of DRVDN[18:17].
29:28	0x0	CFG2TMC_SDIO3CFG_CAL_DRVDN_SLWR. Maps to MSB of DRVUP[26:25].
26:20	0x13	CFG2TMC_SDIO3CFG_CAL_DRVUP. 3.3V 50 ohm driver for Wifi SDIO card.
18:12	0x18	CFG2TMC_SDIO3CFG_CAL_DRVDN. 3.3V 50 ohm driver for removable SD card.
3	0x0	CFG2TMC_SDIO3CFG_SCHMT_EN: SDIO3CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_SDIO3CFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.16 APB_MISC_GP_SPICFGPADCTRL_0

SPICFG Pad Control Register

Offset: 0x8b4 | Read/Write: R/W | Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_SPICFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_SPICFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_SPICFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_SPICFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS

Bit	Reset	Description
5:4	0x3	CFG2TMC_SPICFG_LPMD: SPICFG data pins low power mode select
3	0x0	CFG2TMC_SPICFG_SCHMT_EN: SPICFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_SPICFG_HSM_EN: SPICFG data pins high-speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.17 APB_MISC_GP_UAACFGPADCTRL_0

UAACFG Pad Control Register

Offset: 0x8b8 │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_UAACFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UAACFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_UAACFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_UAACFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_UAACFG_LPMD: UAACFG data pins low power mode select
3	0x0	CFG2TMC_UAACFG_SCHMT_EN: UAACFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UAACFG_HSM_EN: UAACFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.18 APB_MISC_GP_UABCFGPADCTRL_0

UABCFG Pad Control Register

Offset: 0x8bc │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_UABCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UABCFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_UABCFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS

Bit	Reset	Description
16:12	0x9	CFG2TMC_UABCFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_UABCFG_LPMD: UABCFG data pins low power mode select
3	0x0	CFG2TMC_UABCFG_SCHMT_EN: UABCFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UABCFG_HSM_EN: UABCFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.19 APB_MISC_GP_UART2CFGPADCTRL_0

UART2CFG Pad Control Register

Offset: 0x8c0 │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_UART2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UART2CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_UART2CFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_UART2CFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_UART2CFG_LPMD: UART2CFG data pins low power mode select
3	0x0	CFG2TMC_UART2CFG_SCHMT_EN: UART2CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UART2CFG_HSM_EN: UART2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.20 APB_MISC_GP_UART3CFGPADCTRL_0

UART3CFG Pad Control Register

Offset: 0x8c4 │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_UART3CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UART3CFG_CAL_DRVDN_SLWR

Bit	Reset	Description
24:20	0xe	CFG2TMC_UART3CFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_UART3CFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_UART3CFG_LPMD: UART3CFG data pins low power mode select
3	0x0	CFG2TMC_UART3CFG_SCHMT_EN: UART3CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UART3CFG_HSM_EN: UART3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.21 APB_MISC_GP_SDIO1CFGPADCTRL_0

Controls for BDSDMEM pads of SDMMC1

The following calibration codes need to be used for the pad under default condition. In general, the calibration pad will provide the code for the pad. To bypass calibration and provide the default code for the required impedance, these values (in decimal) should be used.

Table 56: DRV Codes

Supply	33 ohms		50 ohms		66 ohms		100 ohms	
	Up	DN	Up	DN	Up	DN	Up	DN
1.8V	70	54	42	32	34	23	19	
3.3V			36	20				

SDIO1CFG Pad Control Register

Offset: 0x8ec │ Read/Write: R/W │ Reset: 0x01318000 (0b0000x0010011x0011000xxxxxxxx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_SDIO1CFG_CAL_DRVUP_SLWF. Maps to MSB of DRVDN[18:17]
29:28	0x0	CFG2TMC_SDIO1CFG_CAL_DRVDN_SLWR. Maps to MSB of DRVUP[26:25]
26:20	0x13	CFG2TMC_SDIO1CFG_CAL_DRVUP. 1.8v 50ohm driver for wifi SDIO card.
18:12	0x18	CFG2TMC_SDIO1CFG_CAL_DRVDN. 1.8v 50ohm driver for wifi SDIO card
3	0x0	CFG2TMC_SDIO1CFG_SCHMT_EN: SDIO1CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_SDIO1CFG_HSM_EN: SDIO1CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.22 APB_MISC_GP_DDCCFGPADCTRL_0

CRTCFG Pad Control Register

DDCCFG Pad Control Register

Offset: 0x8fc │ Read/Write: R/W │ Reset: 0xc160003c (0b1100xxx10110xxx00000xxxxxx1111xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DDCCFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_DDCCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DDCCFG_CAL_DRVUP
16:12	0x0	CFG2TMC_DDCCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DDCCFG_LPMD: Data pins low power mode select
3	0x1	CFG2TMC_DDCCFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x1	CFG2TMC_DDCCFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.23 APB_MISC_GP_GMACFGPADCTRL_0

Controls for BDSDMEMLV pads of SDMMC4

The following calibration codes need to be used for the pad under default condition. In general, the calibration pad will provide the code for the pad. To bypass calibration and provide the default code for the required impedance, these values (in hexadecimal) should be used.

Table 57: DRV Codes (5-bit Code)

Supply	33 ohms		50 ohms	
	Up	DN	Up	DN
1.8V	0x7	0x7	0x2	0x1
1.2V	0xE	0xE	0x5	0x5

GMACFG Pad Control Register

Offset: 0x900 │ Read/Write: R/W │ Reset: 0x01360000 (0b0000xxx10011x11000xxxxxx00xx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_GMACFG_CAL_DRVUP_SLWF. Maps to MSB of DRVDN[18:17].
29:28	0x0	CFG2TMC_GMACFG_CAL_DRVDN_SLWR. Maps to MSB of DRVUP[24:23].
24:20	0x13	CFG2TMC_GMACFG_CAL_DRVUP. 1.8V 50 ohm driver for eMMC.
18:14	0x18	CFG2TMC_GMACFG_CAL_DRVDN. 1.8V 50 ohm driver for eMMC.
7:6	0x0	CFG2TMC_GMACFG_DRV_TYPE. 33-50 ohm driver: 0x1 ; 66-100 ohm driver: 0x0

Bit	Reset	Description
3	0x0	CFG2TMC_GMACFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMACFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.24 APB_MISC_GP_GMECFGPADCTRL_0

GMECFG Pad Control Register

Offset: 0x910 │ Read/Write: R/W │ Reset: 0xf0724030 (0b1111xxxx0111001001xxxxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMECFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMECFG_CAL_DRVDN_SLWR
23:19	0xe	CFG2TMC_GMECFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
18:14	0x9	CFG2TMC_GMECFG_CAL_DRVDN
5:4	0x3	CFG2TMC_GMECFG_LPMD: Data pins low power mode select
3	0x0	CFG2TMC_GMECFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMECFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.25 APB_MISC_GP_GMFCFGPADCTRL_0

GMFCFG Pad Control Register

Offset: 0x914 │ Read/Write: R/W │ Reset: 0xf0724030 (0b1111xxxx0111001001xxxxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMFCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMFCFG_CAL_DRVDN_SLWR
23:19	0xe	CFG2TMC_GMFCFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
18:14	0x9	CFG2TMC_GMFCFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_GMFCFG_LPMD: Data pins low power mode select

Bit	Reset	Description
3	0x0	CFG2TMC_GMFCFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMFCFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.26 APB_MISC_GP_GMGCFGPADCTRL_0

GMGCFG Pad Control Register

Offset: 0x918 │ Read/Write: R/W │ Reset: 0xf0724030 (0b1111xxxx0111001001xxxxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMGCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMGCFG_CAL_DRVDN_SLWR
23:19	0xe	CFG2TMC_GMGCFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
18:14	0x9	CFG2TMC_GMGCFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_GMGCFG_LPMD: Data pins low power mode select
3	0x0	CFG2TMC_GMGCFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMGCFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.27 APB_MISC_GP_GMHCFGPADCTRL_0

GMHCFG Pad Control Register

Offset: 0x91c │ Read/Write: R/W │ Reset: 0xf0724030 (0b1111xxxx0111001001xxxxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMHCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMHCFG_CAL_DRVDN_SLWR
23:19	0xe	CFG2TMC_GMHCFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS

Bit	Reset	Description
18:14	0x9	CFG2TMC_GMHCFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_GMHCFG_LPMD: Data pins low power mode select
3	0x0	CFG2TMC_GMHCFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMHCFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.28 APB_MISC_GP_OWRCFGPADCTRL_0

OWRCFG Pad Control Register

Offset: 0x920 | Read/Write: R/W | Reset: 0xc160003c (0b1100xxx10110xxx00000xxxxxx1111xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_OWRCFG_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_OWRCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_OWRCFG_CAL_DRVUP
16:12	0x0	CFG2TMC_OWRCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_OWRCFG_LPMD: Data pins low power mode select
3	0x1	CFG2TMC_OWRCFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x1	CFG2TMC_OWRCFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.29 APB_MISC_GP_UADCFGPADCTRL_0

UDACFG Pad Control Register

Offset: 0x924 | Read/Write: R/W | Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_UDACFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UDACFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_UDACFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS

Bit	Reset	Description
16:12	0x9	CFG2TMC_UDACFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_UDACFG_LPMD: Data pins low power mode select
3	0x0	CFG2TMC_UDACFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UDACFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.30 APB_MISC_GP_DEV3CFGPADCTRL_0

DEV3CFG Pad Control Register

Offset: 0x92c │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DEV3CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DEV3CFG_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_DEV3CFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x9	CFG2TMC_DEV3CFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_DEV3CFG_LPMD: DEV3CFG data pins low power mode select
3	0x0	CFG2TMC_DEV3CFG_SCHMT_EN: DEV3CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DEV3CFG_HSM_EN: DEV3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.31 APB_MISC_GP_CECCFGPADCTRL_0

CECCFG Pad Control Register

Offset: 0x938 │ Read/Write: R/W │ Reset: 0xf1612030 (0b1111xxx10110xxx10010xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_CECCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_CECCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_CECCFG_CAL_DRVUP

Bit	Reset	Description
16:12	0x12	CFG2TMC_CECCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_CECCFG_LPMD: Data pins low power mode select
3	0x0	CFG2TMC_CECCFG_SCHMT_EN: Data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CECCFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.32 APB_MISC_GP_ATCFG6PADCTRL_0

ATCFG6 Pad Control Register

Offset: 0x994 │ Read/Write: R/W │ Reset: 0x01318000 (0b0000x0010011x0011000xxxx00xx00xx)

Bit	Reset	Description
31:30	0x0	CFG2TMC_ATCFG6_CAL_DRVUP_SLWF
29:28	0x0	CFG2TMC_ATCFG6_CAL_DRVDN_SLWR
26:20	0x13	CFG2TMC_ATCFG6_CAL_DRVUP
18:12	0x18	CFG2TMC_ATCFG6_CAL_DRVDN
7:6	0x0	CFG2TMC_ATCFG6_DRV_TYPE
5:4	0x3	CFG2TMC_ATCFG6_LPMD: ATCFG6 data pins low power mode select
3	0x0	CFG2TMC_ATCFG6_SCHMT_EN: ATCFG6 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG6_HSM_EN: ATCFG6 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.33 APB_MISC_GP_DAP5CFGPADCTRL_0

DAP5CFG Pad Control Register

Offset: 0x998 │ Read/Write: R/W │ Reset: 0xf0503030 (0b1111xxx00101xxx00011xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP5CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP5CFG_CAL_DRVDN_SLWR
24:20	0x5	CFG2TMC_DAP5CFG_CAL_DRVUP
16:12	0x3	CFG2TMC_DAP5CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DAP5CFG_LPMD: DAP5CFG data pins low power mode select
3	0x0	CFG2TMC_DAP5CFG_SCHMT_EN: DAP5CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	CFG2TMC_DAP5CFG_HSM_EN: DAP5CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.34 APB_MISC_GP_USB_VBUS_EN_CFGPADCTRL_0

USB_VBUS_EN_CFG Pad Control Register

Offset: 0x99c │ Read/Write: R/W │ Reset: 0xf1612030 (0b1111xxx10110xxx10010xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_USB_VBUS_EN_CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_USB_VBUS_EN_CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_USB_VBUS_EN_CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_USB_VBUS_EN_CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_USB_VBUS_EN_CFG_LPMD: Data pins low power mode select
3	0x0	CFG2TMC_USB_VBUS_EN_CFG_SCHMT_EN: data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_USB_VBUS_EN_CFG_HSM_EN: Data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.35 APB_MISC_GP_AOCFG3PADCTRL_0

AOCFG3 Pad Control Register

Offset: 0x9a0 │ Read/Write: R/W │ Reset: 0x0000003c (0bxx00xxxxxxxxxx00000xxxxx1111xx)

Bit	Reset	Description
29:28	0x0	CFG2TMC_AOCFG3_CAL_DRVDN_SLWR
16:12	0x0	CFG2TMC_AOCFG3_CAL_DRVDN
5:4	0x3	CFG2TMC_AOCFG3_LPMD: AOCFG3 data pins low power mode select
3	0x1	CFG2TMC_AOCFG3_SCHMT_EN: AOCFG3 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x1	CFG2TMC_AOCFG3_HSM_EN: AOCFG3 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.36 APB_MISC_GP_HVCFG0PADCTRL_0

HVCFG0 Pad Control Register

Offset: 0x9a4 | Read/Write: R/W | Reset: 0x0000003c (0bxx00xxxxxxxxxx00000xxxxx1111xx)

Bit	Reset	Description
29:28	0x0	CFG2TMC_HVCFG0_CAL_DRVDN_SLWR
16:12	0x0	CFG2TMC_HVCFG0_CAL_DRVDN
5:4	0x3	CFG2TMC_HVCFG0_LPMD: HVCFG0 data pins low power mode select
3	0x1	CFG2TMC_HVCFG0_SCHMT_EN: HVCFG0 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x1	CFG2TMC_HVCFG0_HSM_EN: HVCFG0 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.37 APB_MISC_GP_SDIO4CFGPADCTRL_0

SDIO4CFG Pad Control Register

Offset: 0x9a8 | Read/Write: R/W | Reset: 0xc0e0003c (0b1100xxx01110xxx00000xxxxxx1111xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_SDIO4CFG_CAL_DRVUP_SLWF:Default for 1.8V 40 ohm driver
29:28	0x0	CFG2TMC_SDIO4CFG_CAL_DRVDN_SLWR:Default for 1.8V 40 ohm driver
24:20	0xe	CFG2TMC_SDIO4CFG_CAL_DRVUP 22 = V1P8_40OHMS 14 = V2P8_40OHMS 10 = V3P3_40OHMS
16:12	0x0	CFG2TMC_SDIO4CFG_CAL_DRVDN 15 = V1P8_40OHMS 9 = V2P8_40OHMS 6 = V3P3_40OHMS
5:4	0x3	CFG2TMC_SDIO4CFG_LPMD: SDIO4CFG data pins low power mode select
3	0x1	CFG2TMC_SDIO4CFG_SCHMT_EN: SDIO4CFG data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x1	CFG2TMC_SDIO4CFG_HSM_EN: SDIO4CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.4.38 APB_MISC_GP_AOCFG0PADCTRL_0

AOCFG0 Pad Control Register

Offset: 0x9ac │ Read/Write: R/W │ Reset: 0xf0e09030 (0b1111xxx01110xxx01001xxxxxx1100xx)

Bit	Reset	Description
31:30	0x3	CFG2TMC_AOCFG0_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_AOCFG0_CAL_DRVDN_SLWR
24:20	0xe	CFG2TMC_AOCFG0_CAL_DRVUP
16:12	0x9	CFG2TMC_AOCFG0_CAL_DRVDN
5:4	0x3	CFG2TMC_AOCFG0_LPMD: AOCFG0 data pins low power mode select
3	0x0	CFG2TMC_AOCFG0_SCHMT_EN: AOCFG0 data pins Schmitt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_AOCFG0_HSM_EN: AOCFG0 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

22.1.5 Pin Mux Selects

Refer to the MPIO/Pinmux section in this document for details on the Pinmux_aux registers.

22.2 APB DMA Controller

The APB DMA Controller is placed between the AHB Bus and the APB Bus and is a master on both buses.

The APB DMA Controller is used for block data transfers from a source location to the destination location. The source may be DRAM or IRAM, and the destination location could be devices placed on APB Bus; or vice versa. DMA transfers are done without any processor intervention other than register writes needed to program the parameters for a particular transfer, and accesses needed to handle any interrupts.

The APB DMA Controller has 32 fully programmable channels, which can all transfer data concurrently.

22.2.1 Features

Hardware Features

- DMA master for transfers between external/internal memory and peripheral devices on the APB Bus
- Two modes of operation: single transfer (once) or continuous
- Programmable burst sizes of 1, 4, or 8 words
- Maximum transfer size is 1 GB per channel, with the minimum size being one word
- Programmable APB Bus widths of 8, 16, and 32 bits
- Separate AHB and APB start addresses
- Per channel trigger and flow control mechanism support
- Channel to channel trigger support, i.e., ability to link up channels to start at the end of another channel's transfer, allowing scattering/gathering of physical memory.
- Interrupt generation at the completion of channel transfer
- Interrupts per channel can be routed to CPU or AVP
- Ability to hold processor until transfer is done
- Wrap mode supported for all channels in Once mode
- Ping-Pong feature supported in continuous mode
- Round-robin arbitration among channels at burst granularity
- Runs on APB clock. APB Clock generally runs at 1:2:2 or 1:2:3 clock ratios (sclk : hclk : pclk).

Software Features and Notes

- The AHB burst size and the FIFO trigger levels (in the modules) should be programmed so that they do not lead to an overflow/underflow of the FIFO.
- The DMA transfer size should be in multiples of 4 bytes only.
- A channel is programmed with the channel enable bit disabled. The channel enable bit should be set last.
- The Global enable bit is used to pause transfers for some time, and then resumes. This pauses transfers from all channels.
- If a channel is disabled while transfer is in progress, the transfer ends after ongoing burst is completed and an interrupt is generated if enabled.
- If the Channel_Pause bit of a particular channel is enabled during a data transfer, the data transfer is paused only on this channel after the completion of the current burst. It can be resumed by making this bit 0.
- Busy bit is set as soon as DMA channel is enabled and gets cleared after transfer is completed.
- Interrupts are "write 1 to clear".

- An interrupt is generated when the last data is accepted by the AHB slave while writing to the AHB bus and once the last data is placed on APB bus while reading from AHB bus. Note that completion of an AHB write does not guarantee the data is written to DRAM. Refer to the AHB section of this document to find out how to determine data is flushed from AHB to the MC for this master.
- When the address wrap feature is disabled, the addresses are incremented after each word transfer. This helps transfer data from/to contiguous locations.
- Usually, APB clients have a buffer of a fixed size. If the APB addresses were incrementing as in the case mentioned above, there is every possibility the address would cross the address limit of that client and go into the address range of another client. To avoid this, the wrap feature is used.
- With the address wrap feature enabled, the LS nibble of the address wraps back to 0 after completion of the programmed number of words instead of incrementing the address.

  Example: AHB burst size = 4 words, AHB start address = 0x4000_0000, AHB wrap on 32 words

  In this case, the AHB addresses would be:

  0x4000_0000, 0x4000_0004, 0x4000_0008, 0x4000_000C

  …….

  0x4000_0070, 0x4000_0074, 0x4000_0078, 0x4000_007C

  Note: Wrap does not occur in between bursts; it is only after completing a burst that the address wraps around. Wrap is possible for both AHB and APB addresses

- The default wrap around on the APB side is wrapping on 1 word. It prevents unnecessary address switching on the APB.
- The parameters of the channel should be programmed first (for example, base address, wrap-around). The control register of that channel should then be programmed. If the control register is programmed first, the current parameters of the DMA would be considered as the programmed values.
- If a burst of 8 is requested with the address that is not aligned to a 8-word boundary (addr [4:2] not zeros), the DMA does a single burst until it aligns itself to the 8-word boundary and then starts issuing burst requests (on AHB address).
- If a burst of 8 is requested and at any point of time the transfer size goes below 8 (less than 8 words left to transfer), the DMA completes the remaining transfers with a burst of 4 or 1, whichever is possible.
- If a burst of 4 is requested with the address that is not aligned to a 4-word boundary (addr [3:2] not zeros), the DMA does a single burst until it aligns itself to the 4-word boundary.
- If a burst of 4 is requested and at any point of time the transfer size goes below 4 (less than 4 words left to transfer), the DMA completes the remaining transfers with a burst of 1.

22.2.2 Functionality

There are 32 channels in APB DMA. An APB DMA channel can transfer specified portions of data from an AHB address space (can be iRAMs or DRAMs on the MC) to an APB address space. APB DMA follows a simple round robin arbitration scheme, starting with channel 0.

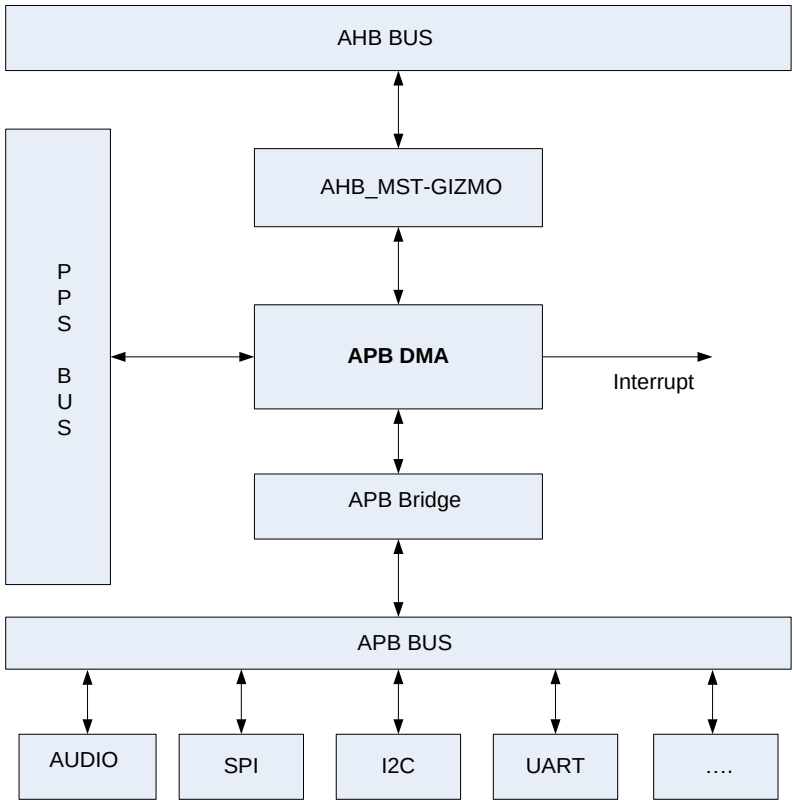
Each channel can have independent burst transfer sizes programmed to one word, four words, or eight words.

Each DMA channel supports:

- Enable bit: one for each channel and one global enable bit.
- Interrupt Enable at the end of transfer with ability to mask or route to desired processor.

- Ability to hold off a processor. The Processor that writes into this bit will be held until transfer is completed.
- Direction bit to determine the direction of transfer--AHB to APB or APB to AHB.
- Trigger and Flow selects. These are addition controls apart from channel enable, on which the transfer depends. Trigger is used to start a channel on some event to start the transfer and flow is used to proceed with every new burst transfer. These events are under either Software or Hardware control.
- Wrap feature wraps the LS nibble of the address back to ‘b0000 instead of incrementing to the next address if the required number of words has been transmitted.
- APB bus width is configurable whereas the AHB bus width is fixed to 32 bits. The APB bus can be 8, 16, or 32 bits wide . For an AHB burst of 8 (burst is always with regard to words) and an APB bus size of 16, there are: 8 words transferred to the APB side, or 16 "halfwords" transferred.
- Double Buffering Mode makes the APB DMA Burst Address reset to AHB Base Address after every even (2nd, 4th, 6th, etc.) APB DMA Transfer. This mode is used only along with continuous mode (once bit 0).
- Separate AHB start address and APB start address.
- Ability to delay the burst rate with the help of a global counter which can be programmed to desired value, which equals number of clocks delay required between each burst.

Figure 61: APB-DMA Block Diagram



22.2.3 Programming Guidelines

Follow these guidelines when programming the APB_DMA Controller:

1. All the registers of a channel should be programmed before the Channel Enable bit is set. To program each register:
   a. Program the AHB Starting Address and APB Starting address in *AHB_PTR and *APB_PTR Registers.

b. Program the required AHB BURST size, WRAP word window size and AHB_DATA_SWAP (byte swapping) option in *AHB_SWQ Register. The AHB BUS WIDTH is fixed to 32-bit Bus.
c. Program the required APB_BUS_WIDTH (as the peripheral), APB_DATA_SWAP (byte swapping) option and WRAP word window size in *APB_SEQ Register.
d. Program the Trigger and most significant 4 bits of DMA Count value in CHANNEL_*_CSRE Register.
e. Program the Interrupt option, Hold Processor option, DMA transfer direction, Transfer Mode, Flow Enable and DMA Count value in CHANNEL_*_CSR Register.
f. Program the Global Enable bit in APB_DMA Command Register. This should be done before programming any register of the channel; otherwise the channel register will not get programmed.
g. Whenever the Channel ENB bit is Enabled, the DMA starts the Data transfer.
h. Each channel status is observed by polling or interrupts status using APB_DMA Status Register.
i. The Tx/Rx Flow/Trigger requestors are programmed in APB-DMA Requestor Assignments Registers.

2. Clearing the Global Enable bit causes the transfers that are in progress to be paused. Setting the Global Enable resumes the transfers. If the Global Enable bit is disabled, then register access is not possible.
3. Setting the channel-Pause bit in CSRE register would pause data transfer on a channel.
4. If channel ENB is disabled independently while a transfer is in progress, the transfer ends after completing any burst sequence that is in progress.
5. Busy bit gets set as soon as a channel is enabled and is cleared after transfer is completed or when the channel is disabled
6. Interrupts are write-1-to-clear i.e., interrupt bit is cleared when the value of write data corresponding to the bit position of the interrupt bit is 1. It is best to read back the status register after the write-1-to-clear.
7. The APB burst size and the FIFO trigger levels (in the modules) need to be programmed such that they do not lead to an overflow/underflow of the FIFO.
8. The AHB wrap around needs to be programmed keeping in mind the address that is programmed in the AHB start address and the burst size.
9. If a burst of 8 is requested with the address that is not aligned to a 8-word boundary, the DMA would do a single burst till it aligns itself to the 8-word boundary and then starts issuing burst requests.
10. If a burst of 8 is requested and at any point of time the transfer size goes below 8, the DMA completes the remaining transfers with a burst of 4 or 1, whichever is possible.
11. If a burst of 4 is requested with the address that is not aligned to a 4-word boundary, the DMA does a single burst till it aligns itself to the 4-word boundary.

If a burst of 4 is requested and at any point of time the transfer size goes below 4, the DMA completes the remaining transfers with a burst of 1.

22.2.4 APB DMA Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

22.2.4.1 APBDMA_COMMAND_0

APB-DMA Command Register

The Global Enable bit in the command register enables the APB DMA. Clearing this bit causes active DMA transfers to be paused. Pending bus transactions (on going burst) will be completed and no new transactions will be initiated. Setting this bit again resumes the transfers. Disabling this bit will disable the channel register access.

Note that the power on reset value for the APB DMA Global Enable is 0. This bit must be written to 1 before any APB DMA transactions can begin.

Offset: 0x0 │Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	GEN: Enables Global APB-DMA 0 = DISABLE 1 = ENABLE

22.2.4.2 APBDMA_STATUS_0

APB-DMA Status Register

The Busy bits in the Status Register indicate which (if any) of the APB DMA channels have active pending APB DMA transfers. Note that APB DMA transfers that are started in continuous (repetitive) mode will have their busy bits active until the enable bit for the APB DMA channel is cleared to 0 (by the software). Read-only Flags set/cleared by hardware.

Offset: 0x4 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	BSY_31: DMA channel31 status 0 = NOT_BUSY 1 = BUSY
30	X	BSY_30: DMA channel30 status 0 = NOT_BUSY 1 = BUSY
29	X	BSY_29: DMA channel29 status 0 = NOT_BUSY 1 = BUSY
28	X	BSY_28: DMA channel28 status 0 = NOT_BUSY 1 = BUSY
27	X	BSY_27: DMA channel27 status 0 = NOT_BUSY 1 = BUSY
26	X	BSY_26: DMA channel26 status 0 = NOT_BUSY 1 = BUSY
25	X	BSY_25: DMA channel25 status 0 = NOT_BUSY 1 = BUSY
24	X	BSY_24: DMA channel24 status 0 = NOT_BUSY 1 = BUSY
23	X	BSY_23: DMA channel23 status 0 = NOT_BUSY 1 = BUSY
22	X	BSY_22: DMA channel22 status 0 = NOT_BUSY 1 = BUSY
21	X	BSY_21: DMA channel21 status 0 = NOT_BUSY 1 = BUSY
20	X	BSY_20: DMA channel20 status 0 = NOT_BUSY

Bit	Reset	Description
		1 = BUSY
19	X	BSY_19: DMA channel19 status 0 = NOT_BUSY 1 = BUSY
18	X	BSY_18: DMA channel18 status 0 = NOT_BUSY 1 = BUSY
17	X	BSY_17: DMA channel17 status 0 = NOT_BUSY 1 = BUSY
16	X	BSY_16: DMA channel16 status 0 = NOT_BUSY 1 = BUSY
15	X	BSY_15: DMA channel15 status 0 = NOT_BUSY 1 = BUSY
14	X	BSY_14: DMA channel14 status 0 = NOT_BUSY 1 = BUSY
13	X	BSY_13: DMA channel13 status 0 = NOT_BUSY 1 = BUSY
12	X	BSY_12: DMA channel12 status 0 = NOT_BUSY 1 = BUSY
11	X	BSY_11: DMA channel11 status 0 = NOT_BUSY 1 = BUSY
10	X	BSY_10: DMA channel10 status 0 = NOT_BUSY 1 = BUSY
9	X	BSY_9: DMA channel9 status 0 = NOT_BUSY 1 = BUSY
8	X	BSY_8: DMA channel8 status 0 = NOT_BUSY 1 = BUSY
7	X	BSY_7: DMA channel7 status 0 = NOT_BUSY 1 = BUSY
6	X	BSY_6: DMA channel6 status 0 = NOT_BUSY 1 = BUSY
5	X	BSY_5: DMA channel5 status 0 = NOT_BUSY 1 = BUSY
4	X	BSY_4: DMA channel4 status 0 = NOT_BUSY 1 = BUSY
3	X	BSY_3: DMA channel3 status 0 = NOT_BUSY 1 = BUSY
2	X	BSY_2: DMA channel2 status 0 = NOT_BUSY

Bit	Reset	Description
		1 = BUSY
1	X	BSY_1: DMA channel1 status 0 = NOT_BUSY 1 = BUSY
0	X	BSY_0: DMA channel0 status 0 = NOT_BUSY 1 = BUSY

22.2.4.3 APBDMA_CNTRL_REG_0

APB-DMA Counter Register

The APB DMA Counter is used to slow down the request rates on some APB DMA channels. The APB DMA Channel Counter Enable register stores bits that configure which (if any) channel(s) should be throttled (bits [31:0] of channel counter enable register) correspond to APB DMA 32 channels.

The APB DMA Counter initial/reload count value is programmed in the APB DMA Counter Register (bits[15:0]).The APB DMA Counter is loaded with this initial/reload count value whenever the APB DMA Counter Register is written, and is re-loaded to this saved initial count value on a burst complete (programmable). The APB DMA Counter value will decrement whenever an APB DMA burst complete, and the current count value is non-zero, and any of the bits [31:16] are set.

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	COUNT_VALUE: DMA COUNT Value.

22.2.4.4 APBDMA_IRQ_STA_CPU_0

APB-DMA CPU IRQ STATUS Register

Gathers all the after-masking CPU directed IRQ status bits

Offset: 0x14 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CH31: Gathers all the after-masking CPU directed IRQ status bits from channel31 0 = DISABLE 1 = ENABLE
30	X	CH30: Gathers all the after-masking CPU directed IRQ status bits from channel30 0 = DISABLE 1 = ENABLE
29	X	CH29: Gathers all the after-masking CPU directed IRQ status bits from channel29 0 = DISABLE 1 = ENABLE
28	X	CH28: Gathers all the after-masking CPU directed IRQ status bits from channel28 0 = DISABLE 1 = ENABLE
27	X	CH27: Gathers all the after-masking CPU directed IRQ status bits from channel27 0 = DISABLE 1 = ENABLE
26	X	CH26: Gathers all the after-masking CPU directed IRQ status bits from channel26 0 = DISABLE 1 = ENABLE
25	X	CH25: Gathers all the after-masking CPU directed IRQ status bits from channel25

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
24	X	CH24: Gathers all the after-masking CPU directed IRQ status bits from channel24 0 = DISABLE 1 = ENABLE
23	X	CH23: Gathers all the after-masking CPU directed IRQ status bits from channel23 0 = DISABLE 1 = ENABLE
22	X	CH22: Gathers all the after-masking CPU directed IRQ status bits from channel22 0 = DISABLE 1 = ENABLE
21	X	CH21: Gathers all the after-masking CPU directed IRQ status bits from channel21 0 = DISABLE 1 = ENABLE
20	X	CH20: Gathers all the after-masking CPU directed IRQ status bits from channel20 0 = DISABLE 1 = ENABLE
19	X	CH19: Gathers all the after-masking CPU directed IRQ status bits from channel19 0 = DISABLE 1 = ENABLE
18	X	CH18: Gathers all the after-masking CPU directed IRQ status bits from channel18 0 = DISABLE 1 = ENABLE
17	X	CH17: Gathers all the after-masking CPU directed IRQ status bits from channel17 0 = DISABLE 1 = ENABLE
16	X	CH16: Gathers all the after-masking CPU directed IRQ status bits from channel16 0 = DISABLE 1 = ENABLE
15	X	CH15: Gathers all the after-masking CPU directed IRQ status bits from channel15 0 = DISABLE 1 = ENABLE
14	X	CH14: Gathers all the after-masking CPU directed IRQ status bits from channel14 0 = DISABLE 1 = ENABLE
13	X	CH13: Gathers all the after-masking CPU directed IRQ status bits from channel13 0 = DISABLE 1 = ENABLE
12	X	CH12: Gathers all the after-masking CPU directed IRQ status bits from channel12 0 = DISABLE 1 = ENABLE
11	X	CH11: Gathers all the after-masking CPU directed IRQ status bits from channel11 0 = DISABLE 1 = ENABLE
10	X	CH10: Gathers all the after-masking CPU directed IRQ status bits from channel10 0 = DISABLE 1 = ENABLE
9	X	CH9: Gathers all the after-masking CPU directed IRQ status bits from channel9 0 = DISABLE 1 = ENABLE
8	X	CH8: Gathers all the after-masking CPU directed IRQ status bits from channel8 0 = DISABLE 1 = ENABLE
7	X	CH7: Gathers all the after-masking CPU directed IRQ status bits from channel7

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
6	X	CH6: Gathers all the after-masking CPU directed IRQ status bits from channel6 0 = DISABLE 1 = ENABLE
5	X	CH5: Gathers all the after-masking CPU directed IRQ status bits from channel5 0 = DISABLE 1 = ENABLE
4	X	CH4: Gathers all the after-masking CPU directed IRQ status bits from channel4 0 = DISABLE 1 = ENABLE
3	X	CH3: Gathers all the after-masking CPU directed IRQ status bits from channel3 0 = DISABLE 1 = ENABLE
2	X	CH2: Gathers all the after-masking CPU directed IRQ status bits from channel2 0 = DISABLE 1 = ENABLE
1	X	CH1: Gathers all the after-masking CPU directed IRQ status bits from channel1 0 = DISABLE 1 = ENABLE
0	X	CH0: Gathers all the after-masking CPU directed IRQ status bits from channel0 0 = DISABLE 1 = ENABLE

22.2.4.5 APBDMA_IRQ_STA_COP_0

APB-DMA COP IRQ STATUS Register

Gathers all the after-masking COP directed IRQ status bits

Offset: 0x18 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CH31: Gathers all the after-masking COP directed IRQ status bits from channel31 0 = DISABLE 1 = ENABLE
30	X	CH30: Gathers all the after-masking COP directed IRQ status bits from channel30 0 = DISABLE 1 = ENABLE
29	X	CH29: Gathers all the after-masking COP directed IRQ status bits from channel29 0 = DISABLE 1 = ENABLE
28	X	CH28: Gathers all the after-masking COP directed IRQ status bits from channel28 0 = DISABLE 1 = ENABLE
27	X	CH27: Gathers all the after-masking COP directed IRQ status bits from channel27 0 = DISABLE 1 = ENABLE
26	X	CH26: Gathers all the after-masking COP directed IRQ status bits from channel26 0 = DISABLE 1 = ENABLE
25	X	CH25: Gathers all the after-masking COP directed IRQ status bits from channel25 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
24	X	CH24: Gathers all the after-masking COP directed IRQ status bits from channel24 0 = DISABLE 1 = ENABLE
23	X	CH23: Gathers all the after-masking COP directed IRQ status bits from channel23 0 = DISABLE 1 = ENABLE
22	X	CH22: Gathers all the after-masking COP directed IRQ status bits from channel22 0 = DISABLE 1 = ENABLE
21	X	CH21: Gathers all the after-masking COP directed IRQ status bits from channel21 0 = DISABLE 1 = ENABLE
20	X	CH20: Gathers all the after-masking COP directed IRQ status bits from channel20 0 = DISABLE 1 = ENABLE
19	X	CH19: Gathers all the after-masking COP directed IRQ status bits from channel19 0 = DISABLE 1 = ENABLE
18	X	CH18: Gathers all the after-masking COP directed IRQ status bits from channel18 0 = DISABLE 1 = ENABLE
17	X	CH17: Gathers all the after-masking COP directed IRQ status bits from channel17 0 = DISABLE 1 = ENABLE
16	X	CH16: Gathers all the after-masking COP directed IRQ status bits from channel16 0 = DISABLE 1 = ENABLE
15	X	CH15: Gathers all the after-masking COP directed IRQ status bits from channel15 0 = DISABLE 1 = ENABLE
14	X	CH14: Gathers all the after-masking COP directed IRQ status bits from channel14 0 = DISABLE 1 = ENABLE
13	X	CH13: Gathers all the after-masking COP directed IRQ status bits from channel13 0 = DISABLE 1 = ENABLE
12	X	CH12: Gathers all the after-masking COP directed IRQ status bits from channel12 0 = DISABLE 1 = ENABLE
11	X	CH11: Gathers all the after-masking COP directed IRQ status bits from channel11 0 = DISABLE 1 = ENABLE
10	X	CH10: Gathers all the after-masking COP directed IRQ status bits from channel10 0 = DISABLE 1 = ENABLE
9	X	CH9: Gathers all the after-masking COP directed IRQ status bits from channel9 0 = DISABLE 1 = ENABLE
8	X	CH8: Gathers all the after-masking COP directed IRQ status bits from channel8 0 = DISABLE 1 = ENABLE
7	X	CH7: Gathers all the after-masking COP directed IRQ status bits from channel7 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	X	CH6: Gathers all the after-masking COP directed IRQ status bits from channel6 0 = DISABLE 1 = ENABLE
5	X	CH5: Gathers all the after-masking COP directed IRQ status bits from channel5 0 = DISABLE 1 = ENABLE
4	X	CH4: Gathers all the after-masking COP directed IRQ status bits from channel4 0 = DISABLE 1 = ENABLE
3	X	CH3: Gathers all the after-masking COP directed IRQ status bits from channel3 0 = DISABLE 1 = ENABLE
2	X	CH2: Gathers all the after-masking COP directed IRQ status bits from channel2 0 = DISABLE 1 = ENABLE
1	X	CH1: Gathers all the after-masking COP directed IRQ status bits from channel1 0 = DISABLE 1 = ENABLE
0	X	CH0: Gathers all the after-masking COP directed IRQ status bits from channel0 0 = DISABLE 1 = ENABLE

22.2.4.6 APBDMA_IRQ_MASK_0

APB-DMA IRQ MASK Register

Allows the IRQ to propagate when enabled, this is the ANDed result of set and clear.

Offset: 0x1c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CH31: Each bit allows the associated channel31 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
30	X	CH30: Each bit allows the associated channel30 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
29	X	CH29: Each bit allows the associated channel29 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
28	X	CH28: Each bit allows the associated channel28 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
27	X	CH27: Each bit allows the associated channel27 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
26	X	CH26: Each bit allows the associated channel26 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
25	X	CH25: Each bit allows the associated channel25 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
24	X	CH24: Each bit allows the associated channel24 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	X	CH23: Each bit allows the associated channel23 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
22	X	CH22: Each bit allows the associated channel22 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
21	X	CH21: Each bit allows the associated channel21 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
20	X	CH20: Each bit allows the associated channel20 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
19	X	CH19: Each bit allows the associated channel19 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
18	X	CH18: Each bit allows the associated channel18 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
17	X	CH17: Each bit allows the associated channel17 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
16	X	CH16: Each bit allows the associated channel16 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
15	X	CH15: Each bit allows the associated channel15 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
14	X	CH14: Each bit allows the associated channel14 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
13	X	CH13: Each bit allows the associated channel13 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
12	X	CH12: Each bit allows the associated channel12 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
11	X	CH11: Each bit allows the associated channel11 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
10	X	CH10: Each bit allows the associated channel10 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
9	X	CH9: Each bit allows the associated channel9 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
8	X	CH8: Each bit allows the associated channel8 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
7	X	CH7: Each bit allows the associated channel7 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
6	X	CH6: Each bit allows the associated channel6 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	X	CH5: Each bit allows the associated channel5 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
4	X	CH4: Each bit allows the associated channel4 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
3	X	CH3: Each bit allows the associated channel3 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
2	X	CH2: Each bit allows the associated channel2 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
1	X	CH1: Each bit allows the associated channel1 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE
0	X	CH0: Each bit allows the associated channel0 IRQ to propagate when '1' 0 = DISABLE 1 = ENABLE

22.2.4.7 APBDMA_IRQ_MASK_SET_0

APB-DMA IRQ MASK SET Register

Offset: 0x20 │ Read/Write: WO │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH31: Sets the Mask Register 0 = DISABLE 1 = ENABLE
30	0x0	CH30: Sets the Mask Register 0 = DISABLE 1 = ENABLE
29	0x0	CH29: Sets the Mask Register 0 = DISABLE 1 = ENABLE
28	0x0	CH28: Sets the Mask Register 0 = DISABLE 1 = ENABLE
27	0x0	CH27: Sets the Mask Register 0 = DISABLE 1 = ENABLE
26	0x0	CH26: Sets the Mask Register 0 = DISABLE 1 = ENABLE
25	0x0	CH25: Sets the Mask Register 0 = DISABLE 1 = ENABLE
24	0x0	CH24: Sets the Mask Register 0 = DISABLE 1 = ENABLE
23	0x0	CH23: Sets the Mask Register 0 = DISABLE 1 = ENABLE
22	0x0	CH22: Sets the Mask Register

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
21	0x0	CH21: Sets the Mask Register 0 = DISABLE 1 = ENABLE
20	0x0	CH20: Sets the Mask Register 0 = DISABLE 1 = ENABLE
19	0x0	CH19: Sets the Mask Register 0 = DISABLE 1 = ENABLE
18	0x0	CH18: Sets the Mask Register 0 = DISABLE 1 = ENABLE
17	0x0	CH17: Sets the Mask Register 0 = DISABLE 1 = ENABLE
16	0x0	CH16: Sets the Mask Register 0 = DISABLE 1 = ENABLE
15	0x0	CH15: Sets the Mask Register 0 = DISABLE 1 = ENABLE
14	0x0	CH14: Sets the Mask Register 0 = DISABLE 1 = ENABLE
13	0x0	CH13: Sets the Mask Register 0 = DISABLE 1 = ENABLE
12	0x0	CH12: Sets the Mask Register 0 = DISABLE 1 = ENABLE
11	0x0	CH11: Sets the Mask Register 0 = DISABLE 1 = ENABLE
10	0x0	CH10: Sets the Mask Register 0 = DISABLE 1 = ENABLE
9	0x0	CH9: Sets the Mask Register 0 = DISABLE 1 = ENABLE
8	0x0	CH8: Sets the Mask Register 0 = DISABLE 1 = ENABLE
7	0x0	CH7: Sets the Mask Register 0 = DISABLE 1 = ENABLE
6	0x0	CH6: Sets the Mask Register 0 = DISABLE 1 = ENABLE
5	0x0	CH5: Sets the Mask Register 0 = DISABLE 1 = ENABLE
4	0x0	CH4: Sets the Mask Register

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
3	0x0	CH3: Sets the Mask Register 0 = DISABLE 1 = ENABLE
2	0x0	CH2: Sets the Mask Register 0 = DISABLE 1 = ENABLE
1	0x0	CH1: Sets the Mask Register 0 = DISABLE 1 = ENABLE
0	0x0	CH0: Sets the Mask Register 0 = DISABLE 1 = ENABLE

22.2.4.8 APBDMA_IRQ_MASK_CLR_0

APB-DMA IRQ MASK CLEAR Register

Offset: 0x24 │ Read/Write: WO │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH31: Clears the Mask Register 0 = DISABLE 1 = ENABLE
30	0x0	CH30: Clears the Mask Register 0 = DISABLE 1 = ENABLE
29	0x0	CH29: Clears the Mask Register 0 = DISABLE 1 = ENABLE
28	0x0	CH28: Clears the Mask Register 0 = DISABLE 1 = ENABLE
27	0x0	CH27: Clears the Mask Register 0 = DISABLE 1 = ENABLE
26	0x0	CH26: Clears the Mask Register 0 = DISABLE 1 = ENABLE
25	0x0	CH25: Clears the Mask Register 0 = DISABLE 1 = ENABLE
24	0x0	CH24: Clears the Mask Register 0 = DISABLE 1 = ENABLE
23	0x0	CH23: Clears the Mask Register 0 = DISABLE 1 = ENABLE
22	0x0	CH22: Clears the Mask Register 0 = DISABLE 1 = ENABLE
21	0x0	CH21: Clears the Mask Register 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
20	0x0	CH20: Clears the Mask Register 0 = DISABLE 1 = ENABLE
19	0x0	CH19: Clears the Mask Register 0 = DISABLE 1 = ENABLE
18	0x0	CH18: Clears the Mask Register 0 = DISABLE 1 = ENABLE
17	0x0	CH17: Clears the Mask Register 0 = DISABLE 1 = ENABLE
16	0x0	CH16: Clears the Mask Register 0 = DISABLE 1 = ENABLE
15	0x0	CH15: Clears the Mask Register 0 = DISABLE 1 = ENABLE
14	0x0	CH14: Clears the Mask Register 0 = DISABLE 1 = ENABLE
13	0x0	CH13: Clears the Mask Register 0 = DISABLE 1 = ENABLE
12	0x0	CH12: Clears the Mask Register 0 = DISABLE 1 = ENABLE
11	0x0	CH11: Clears the Mask Register 0 = DISABLE 1 = ENABLE
10	0x0	CH10: Clears the Mask Register 0 = DISABLE 1 = ENABLE
9	0x0	CH9: Clears the Mask Register 0 = DISABLE 1 = ENABLE
8	0x0	CH8: Clears the Mask Register 0 = DISABLE 1 = ENABLE
7	0x0	CH7: Clears the Mask Register 0 = DISABLE 1 = ENABLE
6	0x0	CH6: Clears the Mask Register 0 = DISABLE 1 = ENABLE
5	0x0	CH5: Clears the Mask Register 0 = DISABLE 1 = ENABLE
4	0x0	CH4: Clears the Mask Register 0 = DISABLE 1 = ENABLE
3	0x0	CH3: Clears the Mask Register 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
2	0x0	CH2: Clears the Mask Register 0 = DISABLE 1 = ENABLE
1	0x0	CH1: Clears the Mask Register 0 = DISABLE 1 = ENABLE
0	0x0	CH0: Clears the Mask Register 0 = DISABLE 1 = ENABLE

22.2.4.9 APBDMA_TRIG_REG_0

APB-DMA Trigger Register

Offset: 0x28 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	TMR2: Trigger select from Timer (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
7	X	TMR1: Trigger select from Timer (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
6	X	XRQ_B: XRQ.B (GPIOB) (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
5	X	XRQ_A: XRQ.A (GPIOA) (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
4	X	SMP_27: Semaphore requests software-initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
3	X	SMP_26: Semaphore requests software-initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
2	X	SMP_25: Semaphore requests software-initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
1	X	SMP_24: Semaphore requests software-initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE

22.2.4.10 APBDMA_CHANNEL_TRIG_REG_0

APB-DMA Channel Trigger Registers

Offset: 0x2c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	APB_31: EOC-31 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
30	X	APB_30: EOC-30 Initiated DMA Request after transfer completion

Bit	Reset	Description
		0 = NOT_ACTIVE 1 = ACTIVE
29	X	APB_29: EOC-29 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
28	X	APB_28: EOC-28 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
27	X	APB_27: EOC-27 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
26	X	APB_26: EOC-26 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
25	X	APB_25: EOC-25 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
24	X	APB_24: EOC-24 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
23	X	APB_23: EOC-23 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
22	X	APB_22: EOC-22 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
21	X	APB_21: EOC-21 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
20	X	APB_20: EOC-20 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
19	X	APB_19: EOC-19 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
18	X	APB_18: EOC-18 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
17	X	APB_17: EOC-17 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
16	X	APB_16: EOC-16 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
15	X	APB_15: EOC-15 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
14	X	APB_14: EOC-14 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
13	X	APB_13: EOC-13 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
12	X	APB_12: EOC-12 Initiated DMA Request after transfer completion

Bit	Reset	Description
		0 = NOT_ACTIVE 1 = ACTIVE
11	X	APB_11: EOC-11 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
10	X	APB_10: EOC-10 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
9	X	APB_9: EOC-9 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
8	X	APB_8: EOC-8 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
7	X	APB_7: EOC-7 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
6	X	APB_6: EOC-6 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
5	X	APB_5: EOC-5 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
4	X	APB_4: EOC-4 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
3	X	APB_3: EOC-3 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
2	X	APB_2: EOC-2 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
1	X	APB_1: EOC-1 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE
0	X	APB_0: EOC-0 Initiated DMA Request after transfer completion 0 = NOT_ACTIVE 1 = ACTIVE

22.2.4.11 APBDMA_DMA_STATUS_0

APB-DMA Interrupt Status

Offset: 0x30 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	ISE_EOC_31: DMA Channel31 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
30	X	ISE_EOC_30: DMA Channel30 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
29	X	ISE_EOC_29: DMA Channel29 Interrupt Status 0 = NOT_ACTIVE

Bit	Reset	Description
		1 = ACTIVE
28	X	ISE_EOC_28: DMA Channel28 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
27	X	ISE_EOC_27: DMA Channel27 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
26	X	ISE_EOC_26: DMA Channel26 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
25	X	ISE_EOC_25: DMA Channel25 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
24	X	ISE_EOC_24: DMA Channel24 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
23	X	ISE_EOC_23: DMA Channel23 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
22	X	ISE_EOC_22: DMA Channel22 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
21	X	ISE_EOC_21: DMA Channel21 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
20	X	ISE_EOC_20: DMA Channel20 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
19	X	ISE_EOC_19: DMA Channel19 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
18	X	ISE_EOC_18: DMA Channel18 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
17	X	ISE_EOC_17: DMA Channel17 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
16	X	ISE_EOC_16: DMA Channel16 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
15	X	ISE_EOC_15: DMA Channel15 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
14	X	ISE_EOC_14: DMA Channel14 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
13	X	ISE_EOC_13: DMA Channel13 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
12	X	ISE_EOC_12: DMA Channel12 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
11	X	ISE_EOC_11: DMA Channel11 Interrupt Status 0 = NOT_ACTIVE

Bit	Reset	Description
		1 = ACTIVE
10	X	ISE_EOC_10: DMA Channel10 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
9	X	ISE_EOC_9: DMA Channel9 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
8	X	ISE_EOC_8: DMA Channel8 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
7	X	ISE_EOC_7: DMA Channel7 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
6	X	ISE_EOC_6: DMA Channel6 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
5	X	ISE_EOC_5: DMA Channel5 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
4	X	ISE_EOC_4: DMA Channel4 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
3	X	ISE_EOC_3: DMA Channel3 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
2	X	ISE_EOC_2: DMA Channel2 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
1	X	ISE_EOC_1: DMA Channel1 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE
0	X	ISE_EOC_0: DMA Channel0 Interrupt Status 0 = NOT_ACTIVE 1 = ACTIVE

22.2.4.12 APBDMA_CHANNEL_EN_REG_0

APB-DMA Channel Counter Enable Registers

Offset: 0x34 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH31_CNT_EN: Enable the Channel31 count 0 = DISABLE 1 = ENABLE
30	0x0	CH30_CNT_EN: Enable the Channel30 count 0 = DISABLE 1 = ENABLE
29	0x0	CH29_CNT_EN: Enable the Channel29 count 0 = DISABLE 1 = ENABLE
28	0x0	CH28_CNT_EN: Enable the Channel28 count 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	0x0	CH27_CNT_EN: Enable the Channel27 count 0 = DISABLE 1 = ENABLE
26	0x0	CH26_CNT_EN: Enable the Channel26 count 0 = DISABLE 1 = ENABLE
25	0x0	CH25_CNT_EN: Enable the Channel25 count 0 = DISABLE 1 = ENABLE
24	0x0	CH24_CNT_EN: Enable the Channel24 count 0 = DISABLE 1 = ENABLE
23	0x0	CH23_CNT_EN: Enable the Channel23 count 0 = DISABLE 1 = ENABLE
22	0x0	CH22_CNT_EN: Enable the Channel22 count 0 = DISABLE 1 = ENABLE
21	0x0	CH21_CNT_EN: Enable the Channel21 count 0 = DISABLE 1 = ENABLE
20	0x0	CH20_CNT_EN: Enable the Channel20 count 0 = DISABLE 1 = ENABLE
19	0x0	CH19_CNT_EN: Enable the Channel19 count 0 = DISABLE 1 = ENABLE
18	0x0	CH18_CNT_EN: Enable the Channel18 count 0 = DISABLE 1 = ENABLE
17	0x0	CH17_CNT_EN: Enable the Channel17 count 0 = DISABLE 1 = ENABLE
16	0x0	CH16_CNT_EN: Enable the Channel16 count 0 = DISABLE 1 = ENABLE
15	0x0	CH15_CNT_EN: Enable the Channel15 count 0 = DISABLE 1 = ENABLE
14	0x0	CH14_CNT_EN: Enable the Channel14 count 0 = DISABLE 1 = ENABLE
13	0x0	CH13_CNT_EN: Enable the Channel13 count 0 = DISABLE 1 = ENABLE
12	0x0	CH12_CNT_EN: Enable the Channel12 count 0 = DISABLE 1 = ENABLE
11	0x0	CH11_CNT_EN: Enable the Channel11 count 0 = DISABLE 1 = ENABLE
10	0x0	CH10_CNT_EN: Enable the Channel10 count 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	0x0	CH9_CNT_EN: Enable the Channel9 count 0 = DISABLE 1 = ENABLE
8	0x0	CH8_CNT_EN: Enable the Channel8 count 0 = DISABLE 1 = ENABLE
7	0x0	CH7_CNT_EN: Enable the Channel7 count 0 = DISABLE 1 = ENABLE
6	0x0	CH6_CNT_EN: Enable the Channel6 count 0 = DISABLE 1 = ENABLE
5	0x0	CH5_CNT_EN: Enable the Channel5 count 0 = DISABLE 1 = ENABLE
4	0x0	CH4_CNT_EN: Enable the Channel4 count 0 = DISABLE 1 = ENABLE
3	0x0	CH3_CNT_EN: Enable the Channel3 count 0 = DISABLE 1 = ENABLE
2	0x0	CH2_CNT_EN: Enable the Channel2 count 0 = DISABLE 1 = ENABLE
1	0x0	CH1_CNT_EN: Enable the Channel1 count 0 = DISABLE 1 = ENABLE
0	0x0	CH0_CNT_EN: Enable the Channel0 count 0 = DISABLE 1 = ENABLE

22.2.4.13 APBDMA_SECURITY_REG_0

Security enables for each channel.

Offset: 0x38 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH_31_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
30	0x0	CH_30_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
29	0x0	CH_29_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
28	0x0	CH_28_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
27	0x0	CH_27_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
26	0x0	CH_26_SECURITY_EN:Enables secure channel 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
25	0x0	CH_25_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
24	0x0	CH_24_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
23	0x0	CH_23_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
22	0x0	CH_22_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
21	0x0	CH_21_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
20	0x0	CH_20_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
19	0x0	CH_19_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
18	0x0	CH_18_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
17	0x0	CH_17_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
16	0x0	CH_16_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
15	0x0	CH_15_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
14	0x0	CH_14_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
13	0x0	CH_13_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
12	0x0	CH_12_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
11	0x0	CH_11_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
10	0x0	CH_10_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
9	0x0	CH_9_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
8	0x0	CH_8_SECURITY_EN:Enables secure channel 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
7	0x0	CH_7_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
6	0x0	CH_6_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
5	0x0	CH_5_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
4	0x0	CH_4_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
3	0x0	CH_3_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
2	0x0	CH_2_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
1	0x0	CH_1_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE
0	0x0	CH_0_SECURITY_EN:Enables secure channel 0 = DISABLE 1 = ENABLE

22.2.4.14 APBDMA_CHANNEL_SWID_0

SWID[0] for each APBDMA channel.

Offset: 0x3c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	CH_31_SWID:SWID for Channel 31
30	0x0	CH_30_SWID:SWID for Channel 30
29	0x0	CH_29_SWID:SWID for Channel 29
28	0x0	CH_28_SWID:SWID for Channel 28
27	0x0	CH_27_SWID:SWID for Channel 27
26	0x0	CH_26_SWID:SWID for Channel 26
25	0x0	CH_25_SWID:SWID for Channel 25
24	0x0	CH_24_SWID:SWID for Channel 24
23	0x0	CH_23_SWID:SWID for Channel 23
22	0x0	CH_22_SWID:SWID for Channel 22
21	0x0	CH_21_SWID:SWID for Channel 21
20	0x0	CH_20_SWID:SWID for Channel 20
19	0x0	CH_19_SWID:SWID for Channel 19
18	0x0	CH_18_SWID:SWID for Channel 18
17	0x0	CH_17_SWID:SWID for Channel 17

Bit	Reset	Description
16	0x0	CH_16_SWID:SWID for Channel 16
15	0x0	CH_15_SWID:SWID for Channel 15
14	0x0	CH_14_SWID:SWID for Channel 14
13	0x0	CH_13_SWID:SWID for Channel 13
12	0x0	CH_12_SWID:SWID for Channel 12
11	0x0	CH_11_SWID:SWID for Channel 11
10	0x0	CH_10_SWID:SWID for Channel 10
9	0x0	CH_9_SWID:SWID for Channel 9
8	0x0	CH_8_SWID:SWID for Channel 8
7	0x0	CH_7_SWID:SWID for Channel 7
6	0x0	CH_6_SWID:SWID for Channel 6
5	0x0	CH_5_SWID:SWID for Channel 5
4	0x0	CH_4_SWID:SWID for Channel 4
3	0x0	CH_3_SWID:SWID for Channel 3
2	0x0	CH_2_SWID:SWID for Channel 2
1	0x0	CH_1_SWID:SWID for Channel 1
0	0x0	CH_0_SWID:SWID for Channel 0

22.2.4.15 APBDMA_SPARE_0

Offset: 0x40 │ Read/Write: R/W │ Reset: 0xffff0000 (0b11111111111111110000000000000000)

Bit	Reset	Description
31:16	0xffff	SPARE_HIGH: Diagnostic workaround bit
15:0	0x0	SPARE_LOW: Diagnostic workaround bit

22.2.5 APB DMA Channel Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

Each of the 32 APB DMA channels has its own set of 8 APB DMA registers. Each channel has 32 bytes of address space. This subsection defines one complete set of APB DMA channel registers. The register spaces per APB DMA channel are listed in the table below.

Table 58: APB DMA Channel Register Mapping

Register Space	Channel Registers
0x0 – 0x1f	Channel 0 APB DMA Channel Registers
0x20 – 0x3f	Channel 1 APB DMA Channel Registers
0x40 – 0x5f	Channel 2 APB DMA Channel Registers
0x60 – 0x7f	Channel 3 APB DMA Channel Registers
0x80 – 0x9f	Channel 4 APB DMA Channel Registers

Register Space	Channel Registers
0xa0 – 0xbf	Channel 5 APB DMA Channel Registers
0xc0 – 0xcf	Channel 6 APB DMA Channel Registers
0xe0 – 0xff	Channel 7 APB DMA Channel Registers
0x100 – 0x11f	Channel 8 APB DMA Channel Registers
0x120 – 0x13f	Channel 9 APB DMA Channel Registers
0x140 – 0x15f	Channel 10 APB DMA Channel Registers
0x160 – 0x17f	Channel 11 APB DMA Channel Registers
0x180 – 0x19f	Channel 12 APB DMA Channel Registers
0x1a0 – 0x1bf	Channel 13 APB DMA Channel Registers
0x1c0 – 0x1df	Channel 14 APB DMA Channel Registers
0x1e0 – 0x1ff	Channel 15 APB DMA Channel Registers
0x200 – 0x21f	Channel 16 APB DMA Channel Registers
0x220 – 0x23f	Channel 17 APB DMA Channel Registers
0x240 – 0x25f	Channel 18 APB DMA Channel Registers
0x260 – 0x27f	Channel 19 APB DMA Channel Registers
0x280 – 0x29f	Channel 20 APB DMA Channel Registers
0x2a0 – 0x2bf	Channel 21 APB DMA Channel Registers
0x2c0 – 0x2df	Channel 22 APB DMA Channel Registers
0x2e0 – 0x2ff	Channel 23 APB DMA Channel Registers
0x300 – 0x31f	Channel 24 APB DMA Channel Registers
0x320 – 0x33f	Channel 25 APB DMA Channel Registers
0x340 – 0x35f	Channel 26 APB DMA Channel Registers
0x360 – 0x37f	Channel 27 APB DMA Channel Registers
0x380 – 0x39f	Channel 28 APB DMA Channel Registers
0x3a0 – 0x3bf	Channel 29 APB DMA Channel Registers
0x3c0 – 0x3df	Channel 30 APB DMA Channel Registers
0x3e0 – 0x3ff	Channel 31 APB DMA Channel Registers

22.2.5.1 APBDMACHAN_CHANNEL_n_CSR_0

APB-DMA-n Control Register

There are 32 APB DMA Channel Control Registers, one per channel (n = 0 through 31).

Writing a 1 to bit [31] of an APB DMA Channel Control Register will initiate the APB DMA transfer. Because this action will depend on some values programmed in the other registers, it is recommended that the registers in the address space in the required APB DMA channel be programmed before writing into the control register.

In "Once" mode, the channel is disabled after the APB DMA transfer has completed. When the channel's transfer completes, an interrupt will be sent if the IE.EOC bit is set. If the transfer requires a trigger or flow, then the corresponding trigger selected by the "TRIG_SEL" or "REQ_SEL" field must become active respectively before the channel can starts its transfer. If both Trigger and Flow bits are set, both conditions must be met before the channel starts each DMA burst.

Offset: 0x0 + (n * 0x20) │ Read/Write: R/W │ Reset: 0x00000000 (0b00000xxxxx00000000000000000000xx)

Bit	Reset	Description
31	0x0	ENB: Enables DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupts when DMA block transfer completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Holds this Processor until DMA block transfer completes 0 = DISABLE 1 = ENABLE
28	0x0	DIR:DMA Transfer Direction. 1 = AHB read to APB write. 0 = APB read to AHB write 0 = AHB_WRITE 1 = AHB_READ
27	0x0	ONCE: Run Once or Run Multiple Mode (Allow Retriggering of this Channel). 1 = Run for One Block Transfer. 0 = Run for Multiple Block Transfers 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers). 1 = Link to DRQ source. 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = APBIF_CH4 7 = APBIF_CH5 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = APBIF_CH6 13 = APBIF_CH7 14 = APBIF_CH8 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = NA20 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = APBIF_CH9 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32-bit word cycles

22.2.5.2 APBDMACHAN_CHANNEL_n_STA_0

APB-DMA-n Status Register

There are 32 APB DMA Status Registers, one per channel (n = 0 through 31).

Offset: 0x4 + (n * 0x20) │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RO	X	BSY: Indicates whether DMA Channel Status is active or not 0 = WAIT 1 = ACTIVE
30	RW	0x0	ISE_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
29	RO	X	HALT: Holding Status of Processor 0 = NO_HALT 1 = HALT
28	RO	X	PING_PONG_STA: If dir = AHB_WRITE : 0 - Ping buffer transfer completed ; 1 - Pong buffer transfer completed. If dir = AHB_READ : 1 - Ping buffer transfer completed ; 0 - Pong buffer transfer completed 0 = PING_INTR_STA 1 = PONG_INTR_STA
27	RO	X	DMA_ACTIVITY: Indicates current DMA channel is transferring data 0 = IDLE 1 = BUSY
26:2	RO	X	COUNT: Current 32-bit word cycles. Flags are set /cleared by hardware.

22.2.5.3 APBDMACHAN_CHANNEL_n_DMA_BYTE_STA_0

APB-DMA-n DMA Byte Status Register

There are 32 APB DMA Byte Status Registers, one per channel (n = 0 through 31).

Offset: 0x8 + (n * 0x20) │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

22.2.5.4 APBDMACHAN_CHANNEL_n_CSRE_0

APB-DMA-n Control-Extended Register

There are 32 APB DMA Control Extended Registers, one per channel (n = 0 through 31).

Offset: 0xc + (n * 0x20) │ Read/Write: R/W │ Reset: 0x00X00000 (0b0xxxxxxxxxxx00000000000000000000)

Bit	R/W	Reset	Description
31	RW	0x0	CHANNEL_PAUSE: When enabled, pauses data transfers on the channel 0 = RESUME 1 = PAUSE
22:20	RO	X	STA_COUNT_HI: Current 32-bit word cycles. Flags are set /cleared by hardware. Most significant 3 bits of STA_0_COUNT
19:14	RW	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24

Bit	R/W	Reset	Description
			2 = SMP25 3 = SMP26 4 = SMP27 5 = XRQ_A 6 = XRQ_B 7 = TMR1 8 = TMR2 9 = APB_0 10 = APB_1 11 = APB_2 12 = APB_3 13 = APB_4 14 = APB_5 15 = APB_6 16 = APB_7 17 = APB_8 18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15 25 = APB_16 26 = APB_17 27 = APB_18 28 = APB_19 29 = APB_20 30 = APB_21 31 = APB_22 32 = APB_23 33 = APB_24 34 = APB_25 35 = APB_26 36 = APB_27 37 = APB_28 38 = APB_29 39 = APB_30 40 = APB_31
13:0	RW	0x0	WCOUNT_HI: Most significant 14 bits of CHANNEL_0_CSR_0_WCOUNT

22.2.5.5 APBDMACHAN_CHANNEL_n_AHB_PTR_0

APB-DMA-n AHB Starting Address Pointer Register

There are 32 APB DMA AHB Starting Address Pointer Registers, one per channel (n = 0 through 31).

Offset: 0x10 + (n * 0x20) │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000xx)

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: Software writes to modify

22.2.5.6 APBDMACHAN_CHANNEL_n_AHB_SEQ_0

APB-DMA-n AHB Address Sequencer Register

There are 32 APB DMA AHB Address Sequencer Registers, one per channel (n = 0 through 31).

Offset: 0x14 + (n * 0x20) │ Read/Write: R/W │ Reset: 0x26000000 (0b00100110xxxx0000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = Send interrupt to COP

Bit	Reset	Description
		1 = CPU 0 = COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width. 0 = 8-bit Bus (RSVD). 1 = 16-bit Bus (RSVD). 2 = 32-bit Bus (default). 3 = 64 bit-Bus (RSVD). 4 = 128-bit Bus (RSVD) 0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	AHB_DATA_SWAP: When enabled, the data going to AHB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] }. 0 = DISABLE 1 = ENABLE
26:24	0x6	AHB_BURST: AHB Burst Size DMA Burst Length (encoded). 4 = 1 Word (1x32bits). 5 = 4 Words (4x32 bits). 6 = 8 Words (8x32bits), default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS
19	0x0	DBL_BUF: 2X Double Buffering Mode (for Run-Multiple Mode with No Wrap Operations). 1 = Reload Base Address for 2X blocks (reload every other time). 0 = Reload Base Address for 1X blocks (default) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS
18:16	0x0	WRAP: AHB Address Wrap: AHB Address wrap-around window. 0=No Wrap (default). 5=Wrap on 512 word window. 1=Wrap on 32 word window. 6=Wrap on 1024 word window. 2=Wrap on 64 word window. 7=Wrap on 2048 word window. 3=Wrap on 128 word window. 4=Wrap on 256 word window 0 = NO_WRAP 1 = WRAP_0N_32WORDS 2 = WRAP_ON_64WORDS 3 = WRAP_ON_128WORDS 4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

22.2.5.7 APBDMACHAN_CHANNEL_n_APB_PTR_0

APB-DMA-n APB Starting Address Pointer Register

There are 32 APB DMA APB Starting Address Pointer Registers, one per channel (n = 0 through 31).

Offset: 0x18 + (n * 0x20) │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000xx)

Bit	Reset	Description
19:2	0x0	APB_BASE: APB-DMA Starting address for APB Bus: APB Base address: Upper 12 bits are fixed at 0x700X:XXXX

22.2.5.8 APBDMACHAN_CHANNEL_n_APB_SEQ_0

APB-DMA-n APB Address Sequencer Assignments

There are 32 APB DMA APB Address Sequencer Assignments Registers, one per channel (n = 0 through 31).

Offset: 0x1c + (n * 0x20) │ Read/Write: R/W │ Reset: 0x20010000 (0bx0100xxxxxxxx001xxxxxxxxxxxxxxx)

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8-bit Bus. 1 = 16-bit Bus. 2 = 32-bit Bus (default) 3 = 64-bit Bus. (RSVD). 4 = 128-bit Bus (RSVD)

Bit	Reset	Description
		0 = BUS_WIDTH_8 1 = BUS_WIDTH_16 2 = BUS_WIDTH_32 3 = BUS_WIDTH_64 4 = BUS_WIDTH_128
27	0x0	APB_DATA_SWAP: When enabled, the data going to the APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] }. 0 = DISABLE 1 = ENABLE
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window. 0 = No Wrap. 1 = Wrap on 1 Word Window (default). 2 = Wrap on 2 Word Window. 3 = Wrap on 4 Word Window. 4 = Wrap on 8 Word Window. 5 = Wrap on 16 Word Window. 6 = Wrap on 32 Word Window. 7 = Wrap on 64 Word Windows (rsvd). 0 = NO_WRAP 1 = WRAP_0N_1WORDS 2 = WRAP_ON_2WORDS 3 = WRAP_ON_4WORDS 4 = WRAP_ON_8WORDS 5 = WRAP_ON_16WORDS 6 = WRAP_ON_32WORDS 7 = WRAP_ON_64WORDS

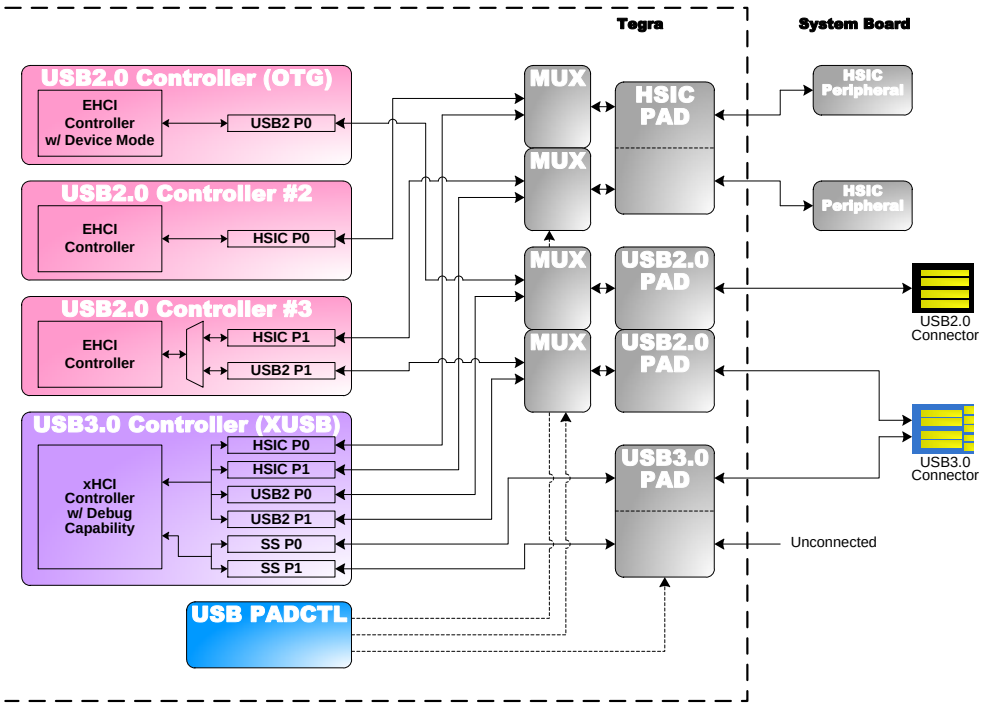
23.0 USB COMPLEX

Note: The ULPI functionality described in this section has been deprecated from Tegra® 4 devices, and is not supported by NVIDIA.

23.1 USB Overview

The Tegra 4 device has four USB controllers (USBOTG, USB2, USB3, and XUSB) and four USB interfaces (UTMIP, HSIC, HSIC1, and UTMIP1). The following diagram shows the relationship between the different USB controllers and interfaces.

Figure 62: Tegra 4 USB Controllers and Interfaces



Below is a summary of the key components in the USB complex:

- USBOTG and UTMIP: This is the primary USB port that supports OTG. USB recovery is also supported on this interface.
- USB2 and UHSIC: This interface allows connection of an HSIC peripheral/host on the PCB board.
- USB3 and UHSIC1: This interface allows connection of an HSIC peripheral/host on the PCB board.
- USB3 and UTMIP1: This interface allows an additional USB port. The primary purpose of this interface is to support host mode of operation.

On power-on-reset, USB2 defaults to unsupported UTMI mode, and USB3 defaults to UTMI mode.

USBOTG supports 16 bidirectional endpoints in device mode, where USB2 and USB3 should never be advised or programmed to support device mode. Endpoint 0 of USBOTG.is always a bidirectional control endpoint. All other endpoints can be programmed as one of Bulk, Interrupt, Isochronous or Control type in either direction. Control endpoints are always bidirectional and hence to program an endpoint as control type, both IN and OUT directions need to be programmed to control type. Conversely, if an endpoint has either IN or OUT direction programmed to be non-control type, the other direction also

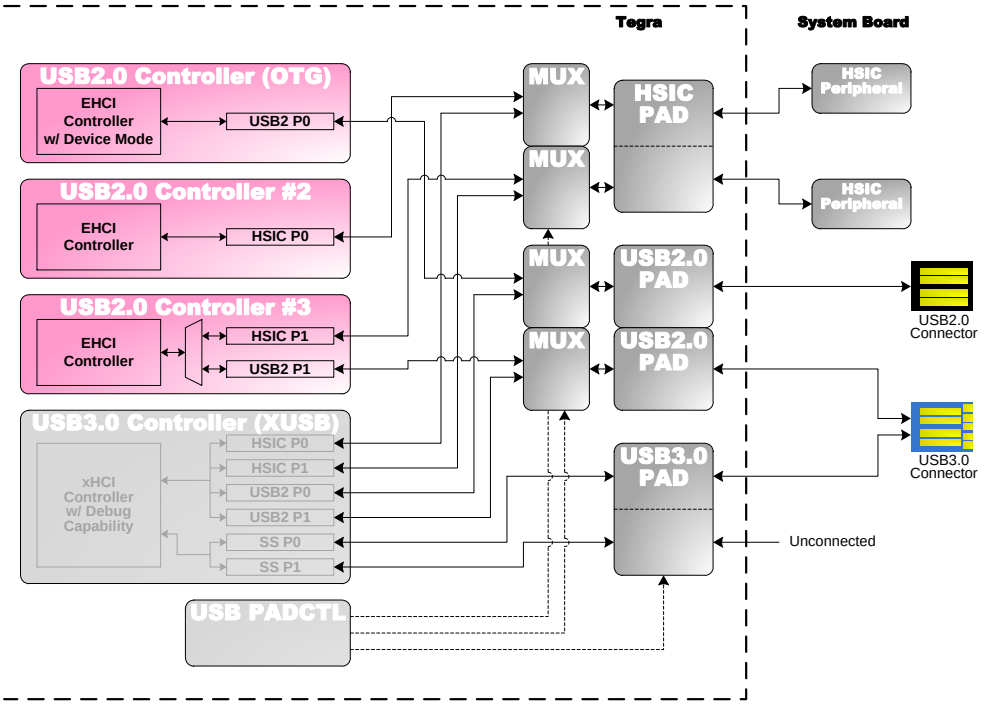
needs to be programmed to be non-control type. The maximum packet size supported on any endpoint is 1024 bytes in high-speed mode, for both device and host modes.

23.2 USB 2.0 Controllers and Interfaces

The USB 2.0 controllers in the Tegra 4 series of processors provide mechanisms for the processor to communicate with a PC as a peripheral or to communicate with USB 2.0 peripherals, such as keyboards, mouse devices, cameras, or storage devices, as a host using regular USB 2.0 ports. The USB 2.0 controllers also provide mechanisms for Tegra 4 devices to communicate with an on-board baseband controller with HSIC or regular interfaces. Each Tegra 4 device provides 3 USB 2.0 controllers with 4 USB interfaces: 2 regular USB ports and 2 HSICs.

The following figure illustrates the connection between USB 2.0 controllers and USB interfaces in a Tegra 4 device, where each USB 2.0 controller can be configured to support 1 USB interface.

Figure 63: Tegra 4 USB 2.0 Controllers and Interfaces



The Tegra 4 device supports peripheral and OTG functionalities with USB 2.0 controller #1, which implements VBUS and ID sensing capabilities and associated interrupt mechanisms to support for device, host, and OTG operations with the regular USB 2.0 port. Tegra 4 devices support battery charging charger detection capabilities and associated interrupt mechanisms that are compliant with USB Battery Charging specification version 1.2 with USB 2.0 controller #1. For both OTG and battery charging, software programming sequences are required to support the operations in response to hardware notifications from USB 2.0 controller #1.

All 3 USB 2.0 controllers support EHCI programming model for scheduling transactions and interface managements as hosts, with each USB 2.0 controller utilizes integrated transaction translator hub to support USB 1.1 transactions when its USB 2.0 interface connects to an USB 1.0 peripheral. USB 2.0 controller #1 supports EHCI-like programming models for scheduling responses to host requests.

Tegra 4 USB 2.0 controllers support L1 and L2 (suspend) link power managements. Tegra 4 USB 2.0 controllers support remote wakeup, wake on connect, wake on disconnect, and wake on overcurrent in all Tegra 4 power states, including deep sleep mode.

23.2.1 USB 2.0 Controller #1

USB 2.0 Controller #1 only connects to USB 2.0 Port #1, which is the primary USB 2.0 port on Tegra 4 devices. USB 2.0 Controller #1 shares the same USB 2.0 Port #1 pins with the USB 3.0 controller. The pinmux must be programmed prior to USB 2.0 Controller #1 using USB 2.0 Port #1.

USB 2.0 Controller #1 supports both USB 2.0 device and USB 2.0 host operations with OTG capabilities. USB recovery is supported only with USB 2.0 Controller #1 and USB 2.0 Port #1.

Wake-up from deep sleep mode is also supported on VBUS and accessory detect (ACCx_DETECT) pins. Power and Ground pins for USB1 and USB3 are shared.

23.2.2 USB 2.0 Controller #2

USB 2.0 Controller #2 must always be configured to Host mode and it can be configured to connect to HSIC Port #1. USB 2.0 Controller #2 can be configured to use HSIC interface that allows connection of an on-board peripheral supporting HSIC interface to Tegra 4 devices. HSIC can be used to support baseband controllers that support the HSIC interface.

23.2.3 USB 2.0 Controller #3

USB 2.0 Controller #3 must always be configured to Host mode, and it can be configured to connect to USB2.0 Port #2 or HSIC Port #2.

USB 2.0 Port

USB 2.0 Controller #3 can be configured to use regular USB 2.0 port. This is the second regular USB 2.0 port on Tegra 4 devices. USB 2.0 Controller #3 shares the same USB 2.0 Port #2 pins with the USB 3.0 controller. The pinmux must be programmed prior to USB 2.0 Controller #1 using USB 2.0 Port #2.

This USB 2.0 port only supports Host mode operations.

HSIC

USB 2.0 Controller #3 can be configured to use the HSIC interface that allows connection of an on-board peripheral supporting an HSIC interface to the Tegra 4 device. HSIC can be used to support baseband controllers that support an HSIC interface.

23.2.4 Interface Restrictions

Regular 2.0 port and HSIC are mutually exclusive for USB 2.0 controller #3, where only one of the interfaces can be used with a given board design.

On power-on-reset, the default interface for USB 2.0 controller #2 is the unsupported UTMIP interface, and the default interface of USB 2.0 controller #3 is regular USB 2.0 port.

23.2.5 AHB Interface

All 3 USB 2.0 controllers adapt AHB interface as masters and slaves for communicating with the other Tegra 4 devices. The following table lists the AHB interface numbers for each controller.

Table 59: AHB Master and Slave Numbers

Controller	AHB Master Number	AHB Slave Number
USB 2.0 Controller #1	6	6
USB 2.0 Controller #2	18	9
USB 2.0 Controller #3	17	11

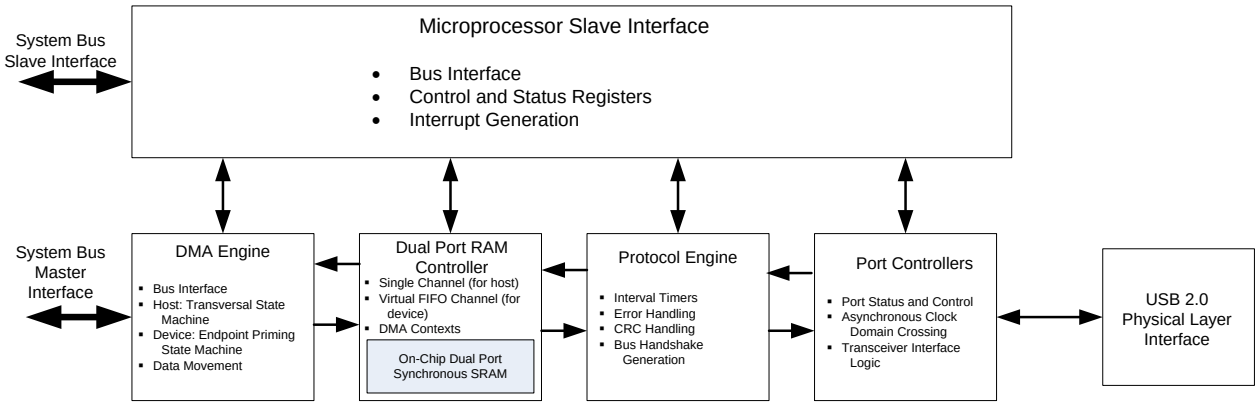
23.3 USB 2.0 Programming Interfaces

All 3 USB 2.0 controllers offer the same functionalities and may be configured to either host or device. However, due to interface restrictions, only USB 2.0 controller #1 should be configured to device mode, while USB 2.0 controller #2 and #3 must always be configured to host mode.

- Host controller registers and data structures are implemented as standard EHCI programming interface.
- Host controller supports USB 1.1 Full and Low speed devices via integrated transaction translator hub through EHCI standard data structures, instead of utilizing companion USB 1.1 host controller.
- Device controller registers and data structures are implemented as extensions to the EHCI programmers interface.

Figure 64 illustrates the control and datapaths block diagram of USB 2.0 controllers.

Figure 64: USB 2.0 Controller Block Diagram



23.3.1 Endpoint Capabilities

USB 2.0 controller #1 supports 16 IN and 16 OUT endpoints in device mode. Endpoint 0 is always a bidirectional control endpoint. All other endpoints can be configured as Bulk, Interrupt, Isochronous or Control endpoints in either direction. Control endpoints are always bidirectional. Hence for a control endpoint, the endpoints in both directions must be configured to be control endpoint. Conversely, if a particular endpoint number has either IN or OUT direction programmed to be a non-control endpoint, the other direction must also be configured as a non-control endpoint.

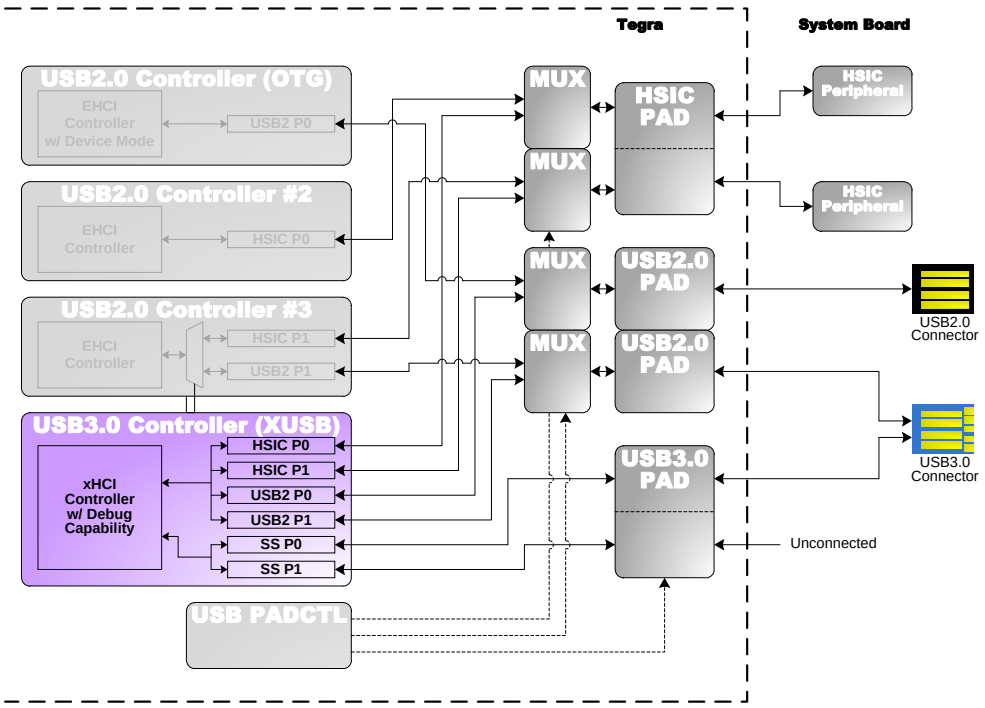
The maximum packet size supported on any endpoint is 1024 bytes in high-speed mode.

23.4 USB 3.0 Controller

The USB 3.0 Controller, also known as XUSB Controller, provides mechanisms for Tegra 4 devices to communicate with USB 2.0 peripherals, such as keyboards, mouse devices, and card readers, and USB 3.0 peripherals, such as cameras and storage devices, as a host using regular USB 3.0 ports. The Tegra 4 device provides 1 USB 3.0 controller that can support up to 2 regular USB 3.0 ports and their companion regular USB 2.0 ports.

The following figure illustrates the connection between USB 3.0 controllers and USB interfaces in Tegra 4 devices, where each USB 3.0 receptacle must encompass 1 USB 3.0 port and 1 USB 2.0 port from the USB 3.0 controller. USB 3.0 Controller shares the same USB 2.0 Port #1 and #2 pins with USB 2.0 controller #1 and #3. The pinmux must be programmed prior to USB 3.0 Controller using USB 2.0 Port #1 or #2.

Figure 65: Tegra 4 USB 3.0 Controller and Interfaces



The USB 3.0 controllers support the xHCI programming model for scheduling transactions and interface managements as a host that natively supports USB 3.0, USB 2.0, and USB 1.1 transactions with its USB 3.0 and USB 2.0 interfaces.

The Tegra 4 USB 3.0 controller supports USB 2.0 L1 and L2 (suspend) link power management and USB 3.0 U1, U2, and U3 (suspend) link power managements. The Tegra 4 USB 3.0 controller supports remote wakeup, wake on connect, wake on disconnect, and wake on overcurrent in all Tegra 4 power states, including deep sleep mode.

USB 2.0 Ports

Each USB 2.0 port operates in USB 2.0 High Speed mode when connecting directly to a USB 2.0 peripheral. Each USB 2.0 port operates in USB 1.1 Full and Low Speed modes when connecting directly to a USB 1.1 peripheral.

All USB 2.0 ports operating in High Speed mode share one High Speed Bus Instance, which means 480 Mb/s theoretical bandwidth is distributed across these ports. All USB 2.0 ports operating in Full or Low Speed modes share one Full/Low Speed Bus Instance, which means 12 Mb/s theoretical bandwidth is distributed across these ports.

USB 2.0 ports of the USB 3.0 controller support software initiated L1 and L2 (suspend) link power management. USB 2.0 ports of the USB 3.0 controller do not support hardware initiated L1 link power management.

USB 3.0 Ports

USB 3.0 ports only operate in USB 3.0 Super Speed mode. All USB 3.0 ports share one Super Speed Bus Instance, which means 5 Gb/s theoretical bandwidth is distributed across these ports.

USB 3.0 ports of the USB 3.0 controller support hardware initiated U1 and U2 link power management as well as software initiate U3 (suspend) link power management.

Falcon Microcontroller

The Tegra 4 USB 3.0 controller integrated an NVIDIA® Falcon microcontroller to perform the following tasks:

- Command ring processing
- Event ring management
- Endpoint scheduling
- Context save and restore

23.4.1 Interface Restrictions

A USB3.0 connector includes both USB2.0 and USB3.0 interface signals. The USB2.0 interface signals of a USB3.0 connector must be assigned to the USB3.0 controller for xHCI specification compliance.

Similarly, if the USB3.0 interface signals are used to connect to a peripheral on the system board, such as a USB3.0 hub, the USB2.0 signals connecting to that peripheral must be assigned to the USB3.0 controller.

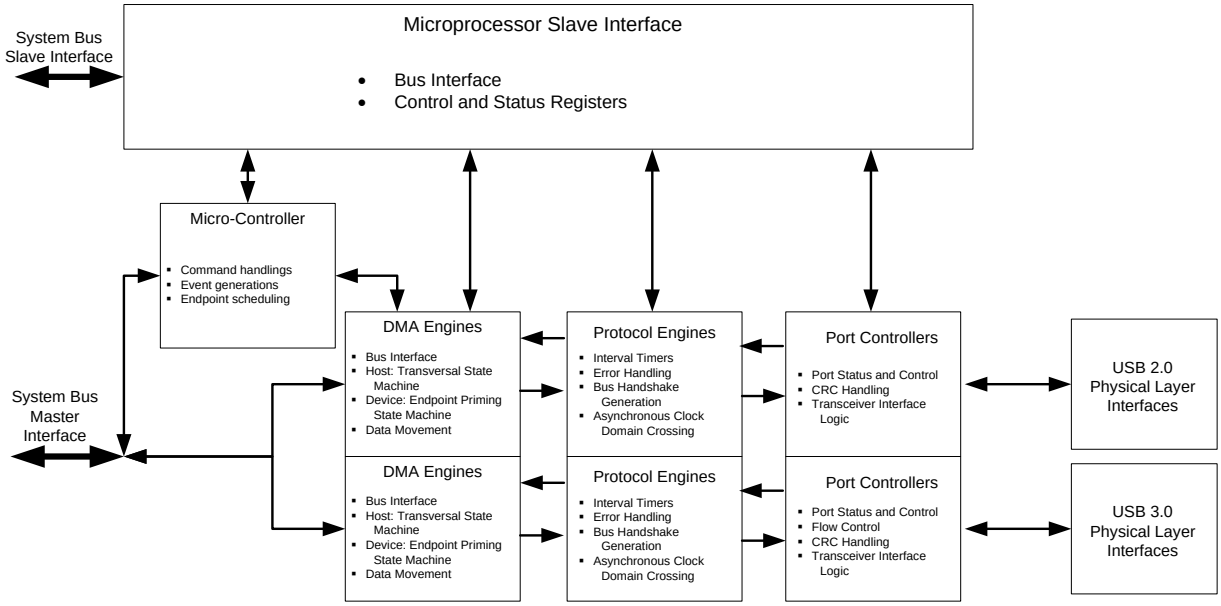
23.4.2 Host and Memory Access Interfaces

USB 3.0 controller adapts the APB interface as its host interface for configuration and memory-mapped I/O register accesses. USB 3.0 controller adapts direct memory as its memory interface for direct memory accesses.

23.5 USB 3.0 Programming Interface

The USB 3.0 Controller supports host functionality with host controller registers and data structures implemented as standard xHCI programming interface. Figure 66 illustrates the control and datapaths block diagram of the USB 3.0 controller.

Figure 66: USB 3.0 Controller Block Diagram



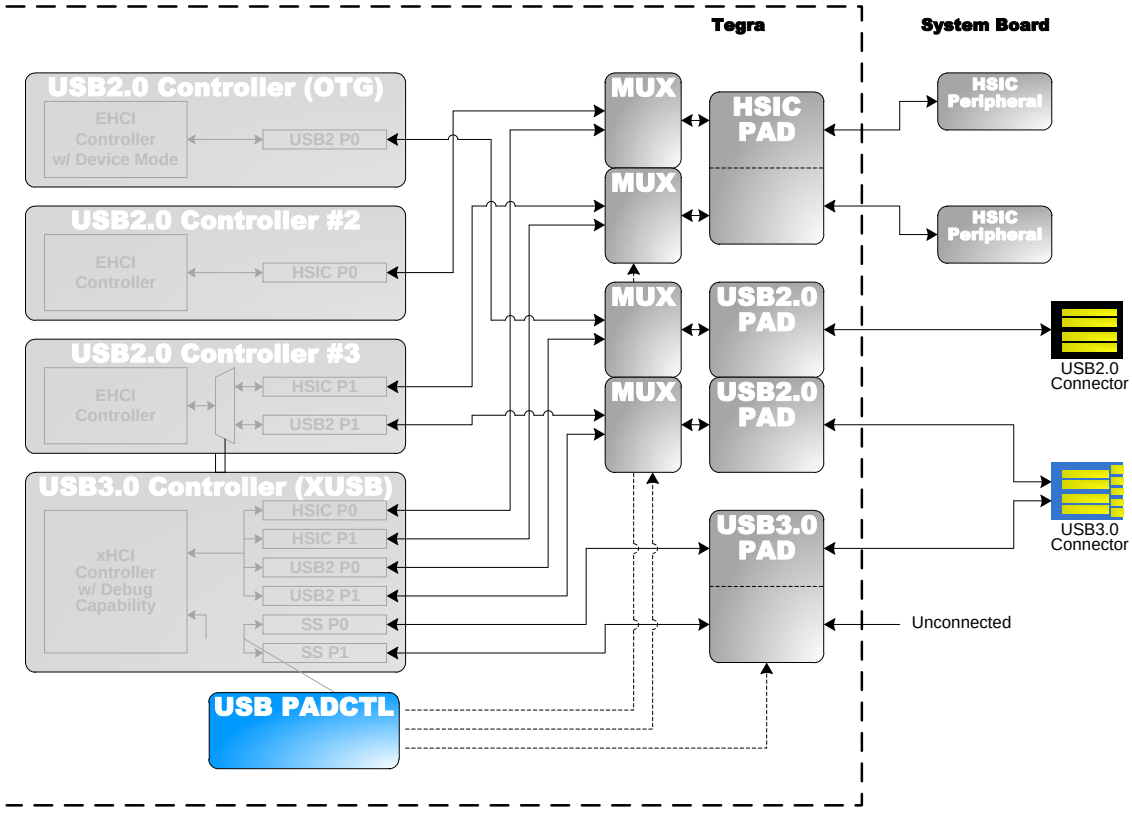
23.6 USB PADCTL

The USB PADCTL in Tegra 4 devices provides the programming interface to assign the USB 2.0 ports to either USB 2.0 controllers or the USB 3.0 controller. The USB PADCTL provides VBUS control and overcurrent mapping for all USB 2.0 controllers and the USB 3.0 controller.

The USB PADCTL also provides the programming interface to configure the pad parameters for the USB 3.0 pad and the USB 2.0 pad for the USB 2.0 port that is assigned to USB 3.0 controller.

The following figure illustrates the connection between USB PADCTL and USB interfaces in a Tegra 4 device.

Figure 67: Tegra 4 USB PADCTL and Interfaces



23.6.1 APB Interface

USB PADCTL adapts the APB interface as its host interface for configuration and memory-mapped I/O register accesses.

23.7 USB 2.0 Programming Guidelines

23.7.1 USB Device/Host Programming

Clock Initialization

After power-on-reset, the USB clocks are disabled. Software can enable the USB clocks by setting CLK_ENB_USBD in CLK_OUT_ENB_L register to ENABLE.

Also, USB stays in reset. So software should bring it out of reset by first setting the SWR_USBD_RST in RST_DEVICES_L register to ENABLE and then setting it to DISABLE.

There are some parameters in UTMIP that need to be programmed before bringing the UTMIP out of reset. Those need to be programmed every time USB is reset. The reset includes SWR_USBD_RST in RST_DEVICES_L register and RST in USB2D_USBCMD register.

After UTMIP is reset, the PHY clock takes some time to come up, so software must wait until the USB_PHY_CLK_VALID bit in the USB_SUSP_CTRL register becomes valid. This bit, if set to 1, also generates an interrupt if RSM_IE is set in the USB_SUSP_CTRL register.

23.7.1.1 ID and VBUS Connection

With Tegra 4 devices, you can use OTG_ID supported by the UTMIP/USB hardware, or use a GPIO for OTG_ID functions. If using an OTG_ID as a GPIO, the GPIO can be sampled to detect the presence of a micro-A or micro-B plug and the sampled value can be written to SW_ID bit in the PHY_VBUS_WAKEUP_ID_0 register. It is recommended that board design connects the OTG_ID pin on the Tegra 4 devices to the ID pin on the micro-AB connector even if the OTG_ID is connected to a GPIO.

Tegra 4 devices have a VBUS pin connected to UTMIP, and the VBUS pin from the connector should be connected to this pin even though VBUS from the connector is connected to a GPIO. It is not required that VBUS from the connector is directly connected to the VBUS pin of the Tegra 4 device. It is required that its voltage is above 3V when VBUS is on.

Both VBUS and ID are available as a wake-up event during DPD/LP0 state. USB bus D+/D- line wake events are also available in DPD/LP0 state. Please consult PMC programming guidelines for further recommendations.

23.7.1.2 Detection of USB Cable Insertion

The USB cable insertion event can be detected by a change in the ID or VBUS pin on the micro-AB connector. Initially, when the cable is not attached, the ID pin on the micro-AB connector is floating and is detected as 1 by the Tegra 4 devices because of the weak pull-up on this pin. This makes the Tegra 4 devices start up as a B-device, even though a cable is not connected (When the ID pin is seen as a 1 by Tegra 4 devices, there can be no-cable or micro-B connected).

When a user attaches the USB cable, one of the following two events occurs:

- The user connects the micro-A end of the cable to Tegra 4 devices. This changes the ID pin to 0. Hence software can detect the ID change and go to A-device mode. Software can then supply power to the USB and places the Tegra 4 devices in host mode.
- The user connects the micro-B (or mini-B) end of the cable to Tegra 4 devices. The other end of the cable can be either micro-A or standard-A and will be connected to a standard host, respectively. For both cases, the Host will start driving power on VBUS and software can detect the cable insertion event by checking the voltage on VBUS has gone above A_Sess_Vld level by reading the A_VBUS_VLD_STS bit in USB_PHY_VBUS_SENSORS register. Once this is seen, it can place the Tegra 4 device in the device mode.

Note: ID change can be detected by enabling the GPIO interrupt.
VBUS change can be detected by enabling the interrupt A_VBUS_VLD_INT_EN in USB_PHY_VBUS_SENSORS register.
USB controller and PHY clocks need not be turned on for this detection.
To detect these events during DPD/LP0 modes, use VBUS/ID wakeup events in the PMC.

23.7.1.3 PHY Clock Control

The PHY clock can be turned off using the SUSP_SET bit in the USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

To turn on the PHY clock, software should write to SUSP_CLR in USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

The current suspend status of the PHY can be checked by reading the bit SUSPENDED in USB_SUSP_CTRL register. If the PHY is placed in suspend, then this bit will be set to 1, else it will be set to 0. Also, if this bit is 1, PHY clock will be turned off and USB_PHY_CLK_VALID in USB_SUSP_CTRL register will be set to 0. If this bit is 0, USB_PHY_CLK_VALID bit can indicate whether the PHY clock is turned on or not.

The PHY clock can be turned on automatically on the events mentioned below. Set only the required wakeup enable bits based on the state of the USB as mentioned below. Do not set unnecessary wakeup enable bits. When the PHY clock wakes up under any such event, make sure that the wakeup enable bits are turned off as soon as possible.

An interrupt is generated whenever the PHY clock is turned on, which can be checked by checking the value of USB_PHY_CLK_VALID in the USB_SUSP_CTRL register to be 1. The interrupt can be enabled/disabled by setting the RSM_IE bit in USB_SUSP_CTRL to 1/0.

Note:	The PLLU_ENABLE bit in PLLU_BASE register should be always set to ENABLE. All the parameters for PLLU in PLLU_BASE and PLLU_MISC register should be set according to the oscillator frequency used. Not doing that will put the USB PHY PLL in a free-running mode, and can result in undesirable side effects. When the USB PHY is suspended, the PCLK should not be stopped. It can be reduced to 32 KHz, however. If the system clock is stopped, then it may not be possible to bring up the system on a wakeup event described below.

There are two cases to consider for controlling the PHY clocks.

Device Mode

The PHY clock can be turned off in two cases:

- When the USB cable is not connected
- When the USB Host puts the USB bus in suspend mode

When the USB cable is not connected, the VBUS signal is 0. In this case, the software needs to turn on the PHY clock on cable insertion. Follow the procedure described above in “Detection of USB Cable Insertion.”

When the USB cable is connected, the PHY clock can only be turned off when the bit SLI in USB2D_USBSTS register is set to 1 when the USB Host puts the USB bus in suspend mode. This bit can be used to generate an interrupt if the bit SLE in USB2D_USBINTR register if set to 1. In this case, software needs to set two bits in the USB_SUSP_CTRL register to turn on the PHY clock on a USB resume or a USB reset event from USB Host: WAKE_ON_DISCON_EN: Wake on Disconnect enable, and WK_RSM_EN - Wake on Resume enable.

If WK_RSM_EN (bit 8) in USB_SUSP_CTRL register is set to 1, the PHY clock can be turned on automatically on receiving USB resume event from USB Host. If WAKE_ON_DISCON_EN in USB_SUSP_CTRL register is set to 1, the PHY clock can be turned on automatically on receiving USB reset event from USB Host or if the USB cable is disconnected. These bits should only be turned on after the PHY clock is turned off. The method described above can be used to interrupt the processor when the PHY clock is turned on. Also, turn off these bits when coming out of suspend.

Host Mode

When a device is not connected, software can set the WKCN bit in USB2D_PORTSC1 register. Make sure that VBUS is also turned on. After this, it can stop the PHY clock by using the method described above. The PHY clock will resume automatically when a device is connected. The method described above can be used to interrupt the processor when the PHY clock is turned on.

With a device connected, software needs to put the USB system under suspend by setting the SUSP bit in the USB2D_PORTSC1 register. It needs to wait until this bit is set to 1 by the controller to make sure that the USB system actually went into suspend. It can enable WK_RSM_EN in the USB_SUSP_CTRL register and WK_DS in the USB2D_PORTSC1 register. Then it can stop the PHY clock as described above. The PHY clock will be turned on automatically if the USB device disconnects or it does a remote wakeup signaling. There will be an interrupt associated with this as described above.

If the software wants to wakeup USB system, then it needs to turn on the PHY clock as described above.

23.7.1.4 Charger Detection 1 (Non-compliant Chargers)

There can be 4 conditions for the different non-compliant chargers: SE1, FS-PU, LS-PU, SE0. You can have some hooks in software so that you can support any non-compliant charger by following the generic procedure below:

1. VBUS detection (by whatever means software does it; it does not matter)

2. Enable USB controller and PHY power and clocks

3. Wait until PHY_CLKVALID = 1.

4. Wait for 10 microseconds more. This gives time for any filtering to pass through so software does not read the wrong values.

5. Read the LS bits [11:10] in the PORTSC register.

- Case 1: LS = 2'b11: connected to charger_1.
- Case 2: LS = 2'b01: connected to charger_2.
- Case 3: LS = 2'b10: connected to charger_3.
- Case 4: LS = 2'b00: connected to either charger_4 or a PC host.

6. For case 1, 2, or 3, enable charging current as per that charger requirement. This could be different for every implementation.

7. For case 4, either you are connected to PC host or a charger_4 (which could be USB compliant charger as well). Go to step 10.

8. You can set the USB circuit on the Tegra 4 device to low-power mode now. Make sure the charger circuit does not change state.

9. VBUS disconnect. Go back to step 1.

10. Check if you are connected to a USB compliant charger by doing the charger detection from VBAT register. If yes, go back to step 8. If no, go to the next step.

11. Try to set the USB controller to device mode and enable by setting it to RUN mode. Wait for SOF.

12. If you do not see SOF after timeout, then you are connected to charger_4. Set the current limit as required for that charger. Go to Step 7.

23.7.1.5 Charger Detection 2 (Compliant Chargers)

This replaces step 10 from the previous section.

1. Make sure UTMIP_PD_CHRG in UTMIP_BAT_CHRG_CFG0 register is set to 0, so that charger detection circuit is on.

   Set the other register bits in UTMIP_BAT_CHRG_CFG0 register to the following values:

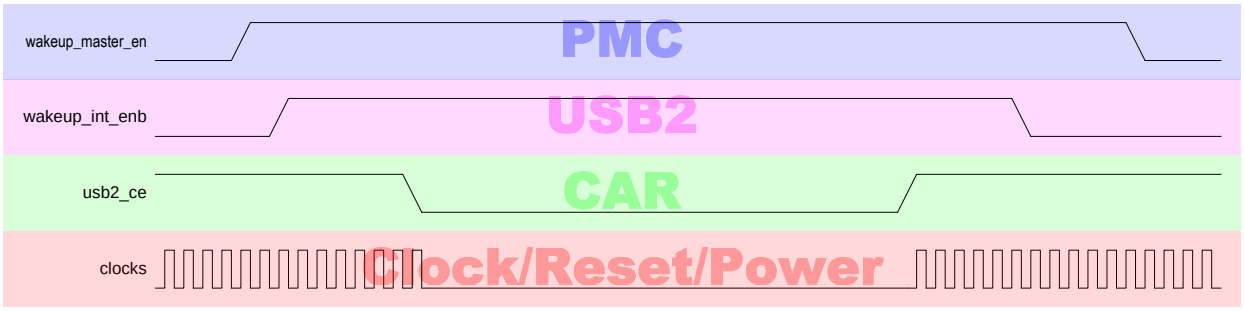
   - UTMIP_OP_I_SRC_EN = 1
2. Wait for about 10 microseconds to allow the battery charger detector to settle.
3. Now check the following bits in the register USB_PHY_VBUS_WAKEUP_ID.
   - VDCD_DET_CHG_DET bit is set, and VDCD_DET_STS indicates 1. Then you are connected to a USB compliant charger.
4. Set the registers bits in UTMIP_BAT_CHRG_CFG0 register to the following values:
   - UTMIP_OP_I_SRC_EN = 0
5. Set the other register bits in UTMIP_BAT_CHRG_CFG0_0 register to the following values:
   - UTMIP_OP_SRC_EN = 1
   - UTMIP_ON_SINK_EN = 1
   - UTMIP_OP_SINK_EN = 0

- UTMIP_ON_SRC_EN = 0

6. Wait for about 10 microseconds to allow the battery charger detector to settle.
7. Now check the following bits in the register USB_PHY_VBUS_WAKEUP_ID.
   - VDAT_DET_CHG_DET bit is set, and VDAT_DET_STS indicates 1. Then you are connected to a USB compliant charger.
8. If it is not set, then we are connected to a PC host.
9. Set the registers bits in UTMIP_BAT_CHRG_CFG0_0 register to the following values:
   - UTMIP_OP_SRC_EN = 0
   - UTMIP_ON_SINK_EN = 0
   - UTMIP_OP_SINK_EN = 0
   - UTMIP_ON_SRC_EN = 0
10. Now normal USB operation can continue if connected to PC host.

23.7.1.6 Controller Clock Gating

USB2 controllers support controller clock gating, where the port wakeup events are detected via the PMC sleepwalk logic. The following figure illustrates the signal sequences to enable USB2 controller clock gating wake event detection before its clocks are gated.



Enable Controller Clock Gating

Step 1

The System Power Management driver programs the PMC USB2.0 sleepwalk logic. This driver should enable the wake events according to the port states and Wake-on-Connect, Wake-on-Disconnect, and Wake-on-Over-Current settings of the USB2 ports accordingly.

> **Note:** The line wakeup event from the PMC is shared between the XUSB and USB2 controllers. There is no per controller masking for the line wake-up event, and when the sleepwalk logic triggers the event, both XUSB and USB2 controllers log the status even when they are not power gated or clock gated. Thus the system power management driver should ensure the wake status is cleared before enabling the sleepwalk logic.

Step 2

The System Power Management driver enables the USB2 controller wakeup interrupt.

Set the following USB2 register bits to '1' to enable the interrupt:

- USB2_UTMIP_PMC_WAKEUP0_0[UTMIP_LINE_WAKEUP_EVENT_INT_ENB] for USB2 #1
- UHSIC_PMC_WAKEUP0_0[UHSIC_LINE_WAKEUP_EVENT_INT_ENB] for USB2 #2

- UTMIP_PMC_WAKEUP0_0[UTMIP_LINE_WAKEUP_EVENT_INT_ENB] for USB2 #3
- UHSIC_PMC_WAKEUP0_0[UHSIC_LINE_WAKEUP_EVENT_INT_ENB] for USB2 #3

Step 3

The System Power Management driver disables the USB2 controller clocks.

Set the following CAR register bits to '1' to disable the clocks:

- CLK_RST_CONTROLLER_CLK_ENB_L_CLR_0[CLR_CLK_ENB_USBD] for USB2 #1
- CLK_RST_CONTROLLER_CLK_ENB_H_CLR_0[CLR_CLK_ENB_USB2] for USB2 #2
- CLK_RST_CONTROLLER_CLK_ENB_H_CLR_0[CLR_CLK_ENB_USB3] for USB2 #3

Step 4

The System Power Management driver puts UTMIPLL to IDDQ if all non-HSIC USB2.0 ports are assigned to USB2 controllers, and all these controllers are in reset or suspend states.

Set the following CAR register bit to '1' to put the PLL to IDDQ:

- CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_OVERRIDE_VALUE]

The System Power Management driver puts UTMIPLL IDDQ to under hardware control when some, but not all, non-HSIC USB2.0 ports are assigned to USB2 controllers, and all of these controllers are in reset or suspend states.

Set the following CAR register bits to '0' to enable UTMIPLL IDDQ hardware control:

- CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_PD_INCLUDE]
- CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_SWCTL]

Disable Controller Clock Gating

Step 1

The System Power Management driver brings UTMIPLL out of to IDDQ when all non-HSIC USB 2.0 ports are assigned to USB2 controllers and any these controllers are exiting reset or suspend states.

Set the following CAR register bit to '0' to put the PLL out of IDDQ:

- CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_OVERRIDE_VALUE]

The System Power Management driver puts UTMIPLL IDDQ to under software control when some, but not all, non-HSIC USB 2.0 ports are assigned to USB2 controllers, and any of these controllers are exiting reset or suspend states.

Set the following CAR register bits to '1' to disable IDDQ HW control:

- CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_PD_INCLUDE]
- CLK_RST_CONTROLLER_UTMIPLL_HW_PWRDN_CFG0_0[UTMIPLL_IDDQ_SWCTL]

Step 2

The System Power Management driver enables the USB2 controller clocks.

Set the following USB2 register bits to '1' to enable the clocks:

- CLK_RST_CONTROLLER_CLK_ENB_L_SET_0[SET_CLK_ENB_USBD] for USB2 #1
- CLK_RST_CONTROLLER_CLK_ENB_H_SET_0[SET_CLK_ENB_USB2] for USB2 #2
- CLK_RST_CONTROLLER_CLK_ENB_H_SET_0[SET_CLK_ENB_USB2] for USB2 #3

Step 3

The System Power Management driver disables the USB controller wake-up interrupt.

Set the following USB2 register bits to '0' to disable the interrupt:

- USB2_UTMIP_PMC_WAKEUP0_0[UTMIP_LINE_WAKEUP_EVENT_INT_ENB] for USB2 #1
- UHSIC_PMC_WAKEUP0_0[UHSIC_LINE_WAKEUP_EVENT_INT_ENB] for USB2 #2
- UTMIP_PMC_WAKEUP0_0[UTMIP_LINE_WAKEUP_EVENT_INT_ENB] for USB2 #3
- UHSIC_PMC_WAKEUP0_0[UHSIC_LINE_WAKEUP_EVENT_INT_ENB] for USB2 #3

Step 4

The System Power Management driver programs the PMC USB 2.0 sleepwalk logic to disable the sleepwalk logic.

23.7.2 LPM Support

Software should program the following register to set the duration the host keeps generating resume signaling when the host initiated exit from L1 based on system and device specific requirements:

USB2_CONTROLLER_USB2D_USBCMD_0.HIRD

The Errata to USB2 LPM renamed the field to BESL and revised the encoding value to extend the duration of host generated resume signaling that allows the device more time to wake up from deeper low power states. The hardware design is not revised to support these errata, where the following software walk-around is required to extend the resume signaling if required.

1. Move PAD out of Suspend

   USB1_IF_USB_SUSP_CTRL_0.USB_SUSP_CLR = SET (1)

2. Fake resume signaling

   USB3_UTMIP_MISC_CFG0_0.UTMIP_DPDM_OBSERVE = 1

   USB3_UTMIP_MISC_CFG0_0.UTMIP_DPDM_OBSERVE_SEL = 0xE

3. Wait required time following the encoding of the HIRD value in the errata
4. Enable hardware to drive the resume signaling

   USB2_CONTROLLER_USB2D_PORTSC1_0.FPR = Resume driven on port (1)

5. Disable fake resume signaling as hardware takes over

   USB3_UTMIP_MISC_CFG0_0.UTMIP_DPDM_OBSERVE = 0

6. Wait hardware finished driving resume signaling by check

   USB2_CONTROLLER_USB2D_PORTSC1_0.FPR = No resume driven on port (0)

7. Wait 50 µs
8. Enable hardware to start sending SOF and executing schedule

   USB2_CONTROLLER_USB2D_USBCMD_0.RS = RUN (1)

23.7.2.1 Software Initiated L1 Support

Software puts the bus to L1 or Suspend (L2) by setting the same SUSP bit in PORTSC1 register. The difference is to put the link to L1, software should first program the following registers:

USB2_CONTROLLER_USB2D_PORTSC1_0.DA = N, Device Address of the attached device

USB2_CONTROLLER_USB2D_HOSTPC1_DEVLC_0. EPLPM = 0

USB2_CONTROLLER_USB2D_PORTSC1_0.SLP = Suspend using L1 (1)

Software should then set the following register to initiate L1 entry

USB2_CONTROLLER_USB2D_PORTSC1_0.SUSP = 1

Software could check the following registers to ensure the link has entered L1

USB2_CONTROLLER_USB2D_PORTSC1_0.SUSP = Port in suspend state (1)

USB2_CONTROLLER_USB2D_PORTSC1_0.SSTS = L1STATE_ENTERED (0)

If L1 cannot be entered successfully, the bus would stay in L0. Hardware would generate port change interrupt when L1 entry failed, where software could check the following registers for the reason or the failed entry. Software could retry the L1 entry in a later time if the reason is NYET_PERIPH, where the peripheral indicates it's not ready to put the bus to L1.

USB2_CONTROLLER_USB2D_PORTSC1_0.SUSP = Port not in suspend state (0)

USB2_CONTROLLER_USB2D_PORTSC1_0.SSTS = NYET_PERIPH (1), or

= L1STATE_NOT_SUPPORTED (2),

= PERIPH_NORESP_ERR (3),

When the bus is in the L1 state, software could set the following register at any time to bring the link back to L0.

USB2_CONTROLLER_USB2D_PORTSC1_0.FPR = Resume driven on port (1)

23.7.2.2 Hardware Initiated L1 Support

Software could enable hardware to put the bus to L1 automatically. Software should first program the following registers, where X is the system specific idle threshold measured in number of SOFs with no activities in between:

USB2_CONTROLLER_USB2D_PORTSC1_0.DA = N, Device Address of the attached device

USB2_CONTROLLER_USB2D_HOSTPC1_DEVLC_0. EPLPM = 0

USB2_CONTROLLER_USB2D_HOSTPC1_DEVLC_0. LPMFRM = X.

Software should then set the following register to enabled hardware initiated L1 entry

USB2_CONTROLLER_USB2D_HOSTPC1_DEVLC_0. LPMX = 1

23.7.3 USB 2.0 Programming Sequence

The USB 2.0 controller supports the UHSIC interface. This interface is directly controlled through the USB2 register space.

23.7.3.1 Clock Initialization

After power-on-reset, USB2 clocks are disabled. Software can enable USB2 clocks by setting CLK_ENB_USB2 in CLK_RST_CLK_OUT_ENB_H register to ENABLE.

Also, USB2 stays in reset at power-on and software should bring it out of reset by first setting the SWR_USB2_RST bit in the RST_DEVICES_H register to ENABLE and then setting it to DISABLE.

After the clocks for USB2 are up, software can program any registers for USB2 required for proper configuration.

PLLU outputs a 480 MHz clock FO_UHSIC for the UHSIC interface. If using any of these interfaces, PLLU should be programmed appropriately.

Any time after USB2 is reset or the UHSIC PHY comes out of suspend, software needs to wait until the UHSIC PHY clock comes up. It can do that by checking the USB_PHY_CLK_VALID bit in USB2_IF_USB_SUSP_CTRL register. If this bit is 1, PHY clocks are up. There are interrupts associated with this bit: if USB_PHY_CLK_VALID_INT_ENB is set to ENABLE, USB_PHY_CLK_VALID_INT_STS will be set to SET whenever the PHY clock becomes valid. Software can write a 1 to USB_PHY_CLK_VALID_INT_STS to clear the interrupt status.

Before doing anything useful on USB2, one of its interfaces needs to be selected and configured.

23.7.3.2 Selection of UHSIC Interface

1. UHSIC I/O pad is in power-down state at power-on. Bring it out of power-down mode by writing PD_BG, PD_TX, PD_TRK, PD_RX, PD_ZI and RPD_DATA, RPD_STROBE fields in UHSIC_PADS_CFG1 register to 0.
2. Bring the tracking circuit out of power-down mode by clearing the PD_TRK bit.
3. Add a 25 µs delay to allow the calibration to complete.
4. Power down the tracking circuit by setting PD_TRK to 1.
5. Hold UHSIC In reset by writing 1 to UHSIC_RESET field in USB2_IF_USB_SUSP_CTRL register.
6. Program the PLLU parameters to enable UHSIC PLLU clock.
7. Select UHSIC interface by writing 1 to UHSIC_PHY_ENB field in USB2_IF_USB_SUSP_CTRL register.
8. Program the configuration parameters for UHSIC.
9. Release reset to UHSIC by writing 0 to UHSIC_RESET field in USB2_IF_USB_SUSP_CTRL register.
10. Set CM field in USB2D_USBMODE register to HOST mode.
11. Program the USB2 controller to use UHSIC PHY by writing 0 to PTS field in USB2_CONTROLLER_1_USB2D_HOSTPC1_DEVLC register.
12. Program the UHSIC_HS_POSTAMBLE_OUTPUT_ENABLE field in the UHSIC_TX_CFG0 register to “1”.
13. Wait until the PHY clock comes up by checking for the USB_PHY_CLK_VALID bit in the USB2_IF_USB_SUSP_CTRL register. Interrupt can be generated as explained above.

There is no VBUS or ID detection required for this interface because the UHSIC does not have these pins.

23.7.3.3 PHY Clock Control

The PHY clock can be turned off by writing 1 to the bit PHCD in USB2D_PORTSC1 register.

> **Note:** There is no need to pulse this bit as opposed to the USB1 (USBOTG) controller.

To turn on the PHY clock, software should write to USB_SUSP_CLR in USB2_IF_USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

Whenever the PHY is placed in suspend, PHY clock will be turned off and USB_PHY_CLK_VALID in USB2_IF_USB_SUSP_CTRL register will be set to 0. Software should make sure that PHY clock shuts down by polling this bit whenever it places the PHY into suspend. Software should make sure that PHY clock shuts down by polling this bit whenever it places the PHY into suspend.

The PHY clock can be turned on automatically on the events mentioned below. Set only the required wakeup enable bits based on the state of the USB as mentioned below. Do not set unnecessary wakeup enable bits. When the PHY clock wakes up under any such event, make sure that the wakeup enable bits are turned off as soon as possible.

To prevent the PHY clock from waking up on unnecessary glitches on the USB pins, software can set the USB_WAKEUP_DEBOUNCE_COUNT field in USB_SUSP_CTRL register to a non-zero value (between 1-7). This will allow the wakeup event to be debounced by the equivalent number of HCLK cycles.

An interrupt is generated whenever PHY is woken up from suspend, which can be verified by checking that the value of USB_WAKEUP_INT_STS in USB_SUSP_CTRL register is 1. The interrupt can be enabled/disabled by setting the bit USB_WAKEUP_INT_ENB in USB_SUSP_CTRL to 1/0.

An interrupt is generated whenever PHY clock is turned on, which can be verified by checking that the value of USB_PHY_CLK_VALID_INT_STS in USB_SUSP_CTRL register is 1. The interrupt can be enabled/disabled by setting the bit USB_PHY_CLK_VALID_INT_ENB in USB_SUSP_CTRL to 1/0.

To clear the interrupts USB_WAKEUP_INT_STS and USB_PHY_CLK_VALID_INT_STS, software can write a 1 to corresponding bits.

Generally, whenever PHY is woken up from suspend, first wakeup event is generated (USB_WAKEUP_INT_STS = 1) followed by PHY clock valid interrupt (USB_PHY_CLK_VALID_INT_STS = 1) when the PHY clock starts up.

When a device is not connected, software can set the WKCN bit in USB2D_PORTSC1 register. Make sure that VBUS is also turned on. After this, it can stop the PHY clock by using the method described above. The PHY clock will resume automatically when a device is connected. The method described above can be used to interrupt the processor when the PHY clock is turned on.

With a device connected, software needs to put the USB system under suspend by setting the bit SUSP in USB2D_PORTSC1 register. It needs to wait until this bit is set to 1 by the controller to make sure that the USB system actually went into suspend. It can enable the bits USB_WAKE_ON_RESUME_EN in USB2_IF_USB_SUSP_CTRL register and WK_DS in USB2D_PORTSC1 register. Then it can stop the PHY clock as described above. The PHY clock will be turned on automatically if the USB device disconnects or it does a remote wakeup signaling. There will be an interrupt associated with this as described above.

If the software wants to wake up the USB system, then it needs to turn on the PHY clock as described above.

23.7.3.4 Bus Signaling Procedure Changes for the UHSIC Interface

To support the UHSIC interface, some changes are required in the bus signaling procedure for USB2 controller (connect and bus reset sequences). These are described in the subsections below:

Host Mode - Bus Connect Sequence

The normal connect sequence assumes that port power has been enabled on a downstream port and the device has signaled a “Connect” on the bus. This will generate a port change interrupt and the software takes action from here. Standard EHCI connect interrupt can be used to detect the device connect.

Host Mode: Bus Reset Sequence

This is not the standard connect but rather a reconnect procedure without physically detaching the USB cable. This operation is demanded by software whenever it is necessary as it is the case when an unrecoverable anomaly takes place. It would then bring the bus, the port and the device into a well-known state.

1. Check if PortReset (PR bit in USB2D_PORTSC1 register) is NOT active – If by any means a bus reset is already taking place it must be stopped before proceeding. This is done by clearing the bit and polling until it goes LOW;
2. Enable the “PortReset” bit - Start the HSIC bus reset on a downstream port by activating the PR bit of the USB2D_PORTSC1 register;
3. Wait until the end of the reset – This wait time is defined by USB 2.0 spec and has the effect of holding the reset signaling on the HSIC bus;

4. Clear the PR bit – Stop the reset signaling by clearing the respective port reset bit;
5. Poll until the reset bit goes LOW – It is advised to guarantee that the reset bit is effectively cleared;
6. Poll until LineState is DPlus – This confirms that the reset signaling has really finished on the HSIC bus and that the USB2 controller is ready to connect again;
7. Proceed with step 2 of the “HOST Bus Connect Sequence” as described above. From here on, the procedure is the same as with an initial connect, instructing the USB2 controller to enter HS directly, skipping speed negotiation.
8. Standard EHCI reset interrupt can be used in Tegra 4 HSIC as native HSIC support in-built.

23.7.4 USB3 Programming Sequence

USB3 controller supports ICUSB and UTMIP3 interfaces. All of these interfaces are directly controlled through USB3 register space.

23.7.4.1 Clock Initialization

After power-on-reset, USB3 clocks are disabled. Software can enable USB3 clocks by setting CLK_ENB_USB3 in ARCLK_RST_CLK_OUT_ENB_H register to ENABLE.

Also, USB3 stays in reset at power-on and software should bring it out of reset by first setting SWR_USB3_RST in RST_DEVICES_H register to ENABLE and then set it to DISABLE.

After the clocks for USB3 are up, software can program any registers for USB3 required for proper configuration.

PLLU outputs a 60 MHz clock FO_ICUSB for ICUSB interface and a 12 MHz clock FO_USB for UTMIP3 interface. If using any of these interfaces, PLLU should be programmed appropriately.

After anytime USB3 is reset, or ICUSB/UTMIP3 PHY comes out of suspend, software needs to wait until ICUSB/UTMIP3 PHY clock comes up. It can do that by checking for USB_PHY_CLK_VALID bit in USB3_IF_USB_SUSP_CTRL register. If this bit is 1, PHY clocks are up, else they are not. There are interrupts associated with this bit: if USB_PHY_CLK_VALID_INT_ENB is set to ENABLE, USB_PHY_CLK_VALID_INT_STS will be set to SET whenever PHY clock becomes valid. Software can write a 1 to USB_PHY_CLK_VALID_INT_STS to clear the interrupt status.

Before doing anything useful on USB2, one of its interfaces needs to be selected and configured.

23.7.4.2 Selection of UTMIP3 Interface

1. UTMIP1 I/O pads are in power-down state at power-on. Bring them out of power-down mode by writing the following fields to 0:
2. FORCE_PD_POWERDOWN, FORCE_PD2_POWERDOWN, FORCE_PDZI_POWERDOWN fields in UTMIP_XCVR_CFG0 register.
3. OTGPD and BIASPD fields in UTMIP_BIAS_CFG0 register.
4. FORCE_PDDISC_POWERDOWN , FORCE_PDCHRP_POWERDOWN, FORCE_PDDR_POWERDOWN field in UTMIP_XCVR_CFG1 register.
5. Hold UTMIP1 PHY in reset by writing UTMIP_RESET bit in USB3_IF_USB_SUSP_CTRL register to 1.
6. Program the PLLU parameters to enable UTMIP3 PLLU clock.
7. Enable UTMIP1 interface by setting UTMIP_PHY_ENB in USB3_IF_USB_SUSP_CTRL register to 1.
8. Program the configuration parameters for UTMIP3.
9. Release reset to UTMIP1 by writing 0 to UTMIP_RESET field in USB3_IF_USB_SUSP_CTRL register.

10. Wait until PHY clock comes up by checking for USB_PHY_CLK_VALID bit in USB3_IF_USB_SUSP_CTRL register. Interrupt can be generated as explained above.
11. Set CM field in USB2D_USBMODE register to HOST mode.
12. Program the USB3 controller to use UTMIP3 PHY by setting the PTS field in USB2_CONTROLLER_2_USB2D_HOSTPC1_DEVLC register to UTMIP (2'b00).

23.7.4.3 Selection of UHSIC Interface

1. UHSIC1 I/O pad is in power-down state at power-on. Bring it out of power-down mode by writing PD_BG, PD_TX, PD_TRK, PD_RX, PD_ZI and RPD_DATA, RPD_STROBE fields in UHSIC_PADS_CFG1 register to 0.
2. Hold UHSIC1 In reset by writing 1 to UHSIC_RESET field in USB3_IF_USB_SUSP_CTRL register.
3. Program the PLLU parameters to enable UHSIC PLLU clock.
4. Select UHSIC interface by writing 1 to UHSIC_PHY_ENB field in USB3_IF_USB_SUSP_CTRL register.
5. Program the configuration parameters for UHSIC.
6. Release reset to UHSIC by writing 0 to UHSIC_RESET field in USB3_IF_USB_SUSP_CTRL register.
7. Set the CM field in the USB2D_USBMODE register to HOST mode.
8. Program the USB3 controller to use UHSIC PHY by writing 0 to PTS field in USB3_CONTROLLER_1_USB2D_HOSTPC1_DEVLC register.
9. Program the UHSIC_HS_POSTAMBLE_OUTPUT_ENABLE field in the UHSIC_TX_CFG0 register to "1".
10. Wait until the PHY clock comes up by checking for USB_PHY_CLK_VALID bit in USB3_IF_USB_SUSP_CTRL register. Interrupt can be generated as explained above.
11. There is no VBUS or ID detection required for this interface because these pins do not exist for UHSIC.

23.7.4.4 Detection of USB Cable Insertion

The USB cable insertion event can be detected by a change in the ID or VBUS pin on the micro-AB connector. Initially, when the cable is not attached, the ID pin on the micro-AB connector is floating and is detected as 1 by Tegra 4 devices because of the weak pull-up on this pin. This makes a Tegra 4 device start up as a B-device, even though a cable is not connected (when the ID pin is seen as a 1 by a Tegra 4 device, there can be no cable or micro-B connected).

When a user attaches the USB cable, one of the following two events occur:

- User connects micro-A end of the cable to the Tegra 4 device. This changes the ID pin to 0. Hence the software can detect the ID change and go to A-device mode. Software can then supply power to the USB and place the Tegra 4 device in host mode.
- User connects micro-B (or mini-B) end of the cable to the Tegra 4 device. The other end of the cable can be either micro-A or standard-A and will be connected to a standard host, respectively.

For both cases, the Host will start driving power on VBUS and software can detect the cable insertion event by checking the VBUS status on corresponding GPIO. Once this is seen, it can place the Tegra 4 device in the device mode.

Note: Both ID and VBUS changes can be detected by enabling the corresponding GPIO interrupts.
USB controller/PHY clocks do not need to be turned on for this detection as this is done using GPIOs.
To detect these events during DPD/LP0 modes, use VBUS/ID wakeup events in the PMC.

23.7.4.5 PHY Clock Control

The PHY clock can be turned off by writing 1 to the PHCD bit in the USB2D_PORTSC1 register.

Note:	There is no need to pulse this bit as opposed to the USB1 (USBOTG) controller.

To turn on the PHY clock, software should write to SUSP_CLR in USB3_IF_USB_SUSP_CTRL register. This bit must be pulsed by first writing a 1 and then writing a 0 to the bit.

The current suspend status of the PHY and thereby whether the PHY clock is running can be checked by reading the bit USB_PHY_CLK_VALID in USB_SUSP_CTRL register. If the PHY is placed in suspend, then this bit will be set to 0, else it will be set to 1. Software should make sure that PHY clock shuts down by polling this bit whenever it places the PHY into suspend.

The PHY clock can be turned on automatically on the events mentioned below. Set only the required wakeup enable bits based on the state of the USB as mentioned below. Do not set unnecessary wakeup enable bits. When the PHY clock wakes up under any such event, make sure that the wakeup enable bits are turned off as soon as possible.

To prevent the PHY clock from waking up on unnecessary glitches on the USB pins, software can set the USB_WAKEUP_DEBOUNCE_COUNT field in USB_SUSP_CTRL register to a non-zero value (between 1-7). This will allow the wakeup event to be debounced by the equivalent number of HCLK cycles.

An interrupt is generated whenever the PHY is woken up from suspend, which can be verified by checking that the value of USB_WAKEUP_INT_STS in USB_SUSP_CTRL register is 1. The interrupt can be enabled/disabled by setting the bit USB_WAKEUP_INT_ENB in USB_SUSP_CTRL to 1/0.

An interrupt is generated whenever PHY clock is turned on, which can be verified by checking that the value of USB_PHY_CLK_VALID_INT_STS in USB_SUSP_CTRL register is 1. The interrupt can be enabled/disabled by setting the bit USB_PHY_CLK_VALID_INT_ENB in USB_SUSP_CTRL to 1/0.

To clear the interrupts USB_WAKEUP_INT_STS and USB_PHY_CLK_VALID_INT_STS, software can write a 1 to the corresponding bits.

Generally, whenever PHY is woken up from suspend, first wakeup event is generated (USB_WAKEUP_INT_STS = 1) followed by PHY clock valid interrupt (USB_PHY_CLK_VALID_INT_STS = 1) when PHY clock starts up.

Note:	The PLLU_ENABLE bit in PLLU_BASE register should be always set to ENABLE. All the parameters for PLLU in PLLU_BASE and PLLU_MISC register should be set according to the oscillator frequency used. Not doing that will put the USB PHY PLL in a free-running mode, and can result in undesirable side-effects. When the USB PHY is suspended, the USB3 clock should not be stopped. It can be reduced to 32 KHz, however. If the system clock is stopped, then it may not be possible to bring up the system on a wakeup event described below. When USB3 is in suspend, AHB clocks to USB3 controller are turned off, and hence there should be no register access to USB2_CONTROLLER_2_* registers. All registers marked as ARUSB3_IF_* are accessible, and can be used to resume the UTMIP3 PHY clocks.

When a device is not connected, software can set the WKCN bit in USB2D_PORTSC1 register. Make sure that VBUS is also turned on. After this, it can stop the PHY clock by using the method described above. The PHY clock will resume automatically when a device is connected. The method described above can be used to interrupt the processor when the PHY clock is turned on.

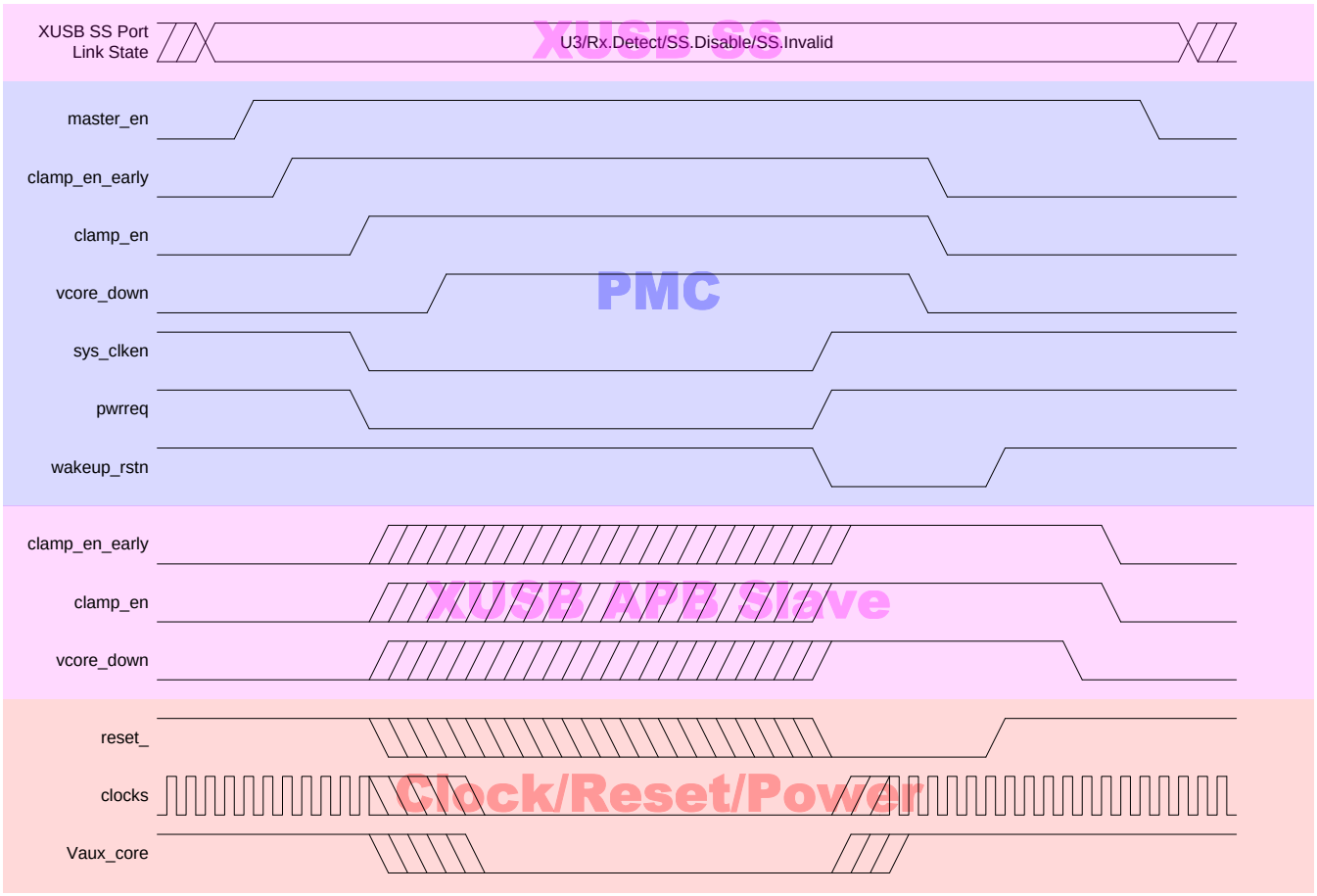
With a device connected, software needs to put the USB system under suspend by setting the bit SUSP in USB2D_PORTSC1 register. It needs to wait until this bit is set to 1 by the controller to make sure that the USB system actually went into suspend. It can enable the bits USB_WAKE_ON_RESUME_EN in USB_SUSP_CTRL register and WK_DS in USB2D_PORTSC1 register. Then it can stop the PHY clock as described above. The PHY clock will be turned on automatically if the USB device disconnects or it does a remote wakeup signaling. There will be an interrupt associated with this as described above.

If the software wants to wake up the USB system, it needs to turn on the PHY clock as described above.

23.7.5 XUSB Controller Power Gating

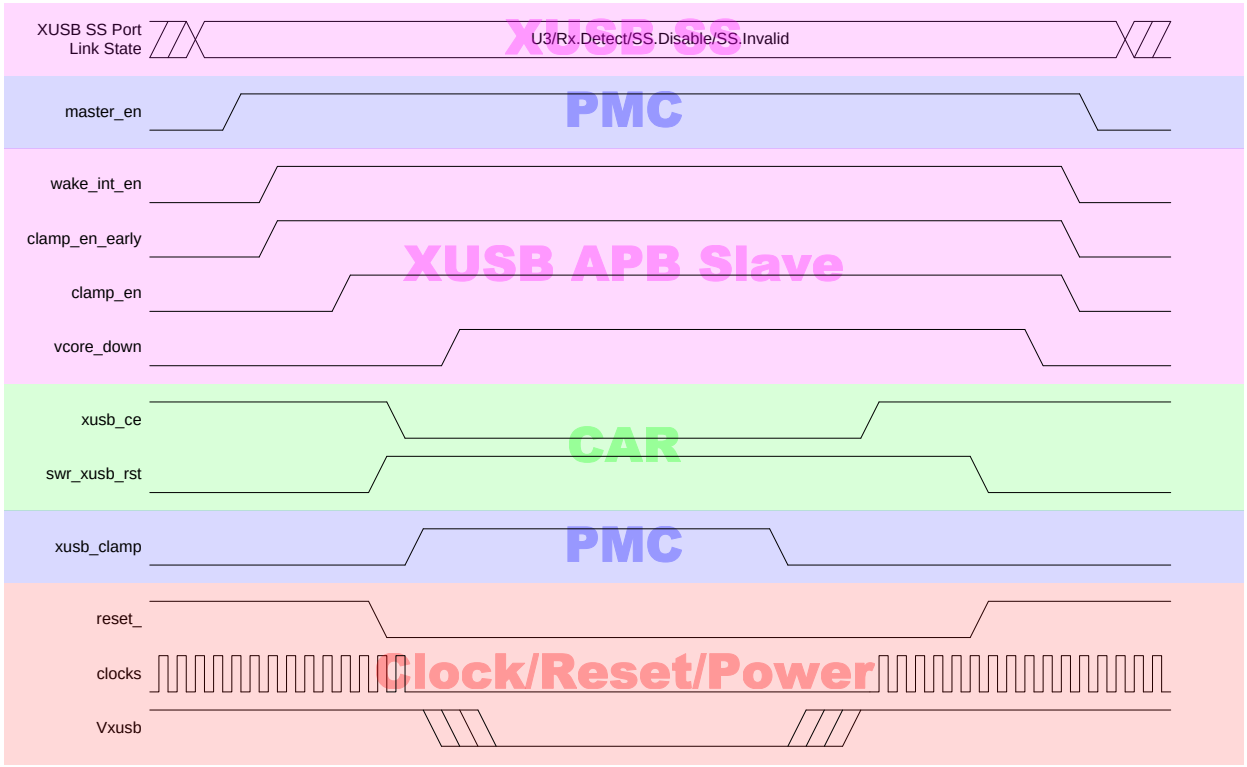
The following figure illustrates the signal sequences from the PMC that are used by XUSB SS Wake logic to enable its wake event detection.

LP0



PMC_1: APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UHSIC_MASTER_ENABLE_P0]
APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P0]
APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0[UTMIP_MASTER_ENABLE_P2]
APBDEV_PMC_UTMIP_UHSIC2_SLEEP_CFG_0[UHSIC_MASTER_ENABLE_P1]
PMC_2: Sequence driven by PMC LP0 state machine

The next figure illustrates the signal sequences from the XUSB PADCTL block that are required to follow the same sequence as from PMC so XUSB SS Wake logic can enable its wake event detection.



XUSB has three partitions that can be selectively power-gated with the following use cases, where unlisted combinations are not supported:

XUSBC (Host/USB2.0)	XUSBB (Device)	XUSBA (SuperSpeed)	Use Case Descriptions
Powered	Powered	Powered	Both Host and Device modes in normal operations, with SuperSpeed links operational.
Powered	Powered	Power Gated	Both Host and Device modes in normal operations, with SuperSpeed links in low power states. SuperSpeed link wake events report through XUSB PADCTL.
Powered	Power Gated	Powered	Host mode in normal operations, with SuperSpeed links operational. Device mode power gated. Device mode SuperSpeed and USB2.0 link wake events report through XUSB PADCTL.
Powered	Power Gated	Power Gated	Host mode in normal operations, with SuperSpeed links in low power states. Device mode power gated. Host Mode SuperSpeed, and Device mode SuperSpeed and USB2.0 link wake events report through XUSB PADCTL.
Power Gated	Power Gated	Power Gated	Both Host and Device modes power gated, with SuperSpeed links in low power states. All Host and Device modes link wake events report through XUSB PADCTL.

23.7.5.1 All Partitions ELPG Entry

Step 1: xHCI driver performs context save operation. xHCI PEP driver performs XUSB_HOST specific context save operation. In case the SuperSpeed partition has already be power gated, xHCI FW and XUSB device mode driver should not save the

context of the SuperSpeed partition again. XUSB device mode driver performs XUSB_DEVICE specific context save operations. xHCI PEP driver and XUSB device mode driver performs XUSB IPFS specific context save operation.

- Read and store the value of the following registers:

  XUSB_{HOST,DEV}_MSI_BAR_SZ_0

  XUSB_{HOST,DEV}_MSI_AXI_BAR_ST_0

  XUSB_{HOST,DEV}_MSI_FPCI_BAR_ST_0

  XUSB_{HOST,DEV}_MSI_VEC0_0

  XUSB_{HOST,DEV}_MSI_EN_VEC0_0

  XUSB_{HOST,DEV}_FPCI_ERROR_MASKS_0

  XUSB_{HOST,DEV}_INTR_MASK_0

  XUSB_{HOST,DEV}_IPFS_INTR_ENABLE_0

  XUSB_{HOST,DEV}_UFPCI_CONFIG_0

  XUSB_{HOST,DEV}_CLKGATE_HYSTERESIS_0

  XUSB_{HOST,DEV}_XUSB_HOST_MCCIF_FIFOCTRL_0

Step 2: System Power Management driver programs the PMC USB2.0 sleepwalk logic for ports assigned to XUSB host mode and XUSB device mode.

System Power Management driver should enable the wake events according to the port states and Wake-on-Connect, Wake-on-Disconnect, and Wake-on-Over-Current settings of the USB2 ports accordingly.

> **Note:** The line wakeup event from PMC is shared between XUSB and USB2 controller. There is no per controller masking for the line wakeup event, and when the sleepwalk logic triggers the event, both XUSB and USB2 will log the status even when they are not power gated or clock gated. Thus the system power management driver should ensure the wake status is cleared before enabling the sleepwalk logic.

Step 3: xHCI PEP driver and XUSB device mode driver enable the XUSB wakeup interrupts for the SuperSpeed and USB2.0 ports assigned to host and device.

- Write ‘1’ to the following XUSB PADCTL register bits to clear the interrupt status.

  XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT0_WAKEUP_EVENT]

  XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT1_WAKEUP_EVENT]

  XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT0_WAKEUP_EVENT]

  XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT1_WAKEUP_EVENT]

  XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT0_WAKEUP_EVENT]

  XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT1_WAKEUP_EVENT]

- Set the following XUSB PADCTL register bits to ‘1’ to enable the interrupt.

  XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT0_WAKE_INTERRUPT_ENABLE]

  XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT1_WAKE_INTERRUPT_ENABLE]

  XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT0_WAKE_INTERRUPT_ENABLE]

  XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT1_WAKE_INTERRUPT_ENABLE]

  XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT0_WAKE_INTERRUPT_ENABLE]

XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT1_WAKE_INTERRUPT_ENABLE]

> **Note:** The interrupt status bit could also be set due to previous wake events reported for USB2 controllers, thus the status should be cleared before enabling the interrupt.

Step 4: xHCI PEP driver and XUSB device mode driver initiate the signal sequence to enable the XUSB SSwake detection logic for the SuperSpeed ports assigned to host and device.

- Write '1' to the following XUSB PADCTL register bit to assert the clamp_en_early signal.

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN_EARLY]

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_CLAMP_EN_EARLY]

- Write '1' to the following XUSB PADCTL register bit to assert the clamp_en signal.

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN]

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_CLAMP_EN]

- Wait 250 µs

> **Note:** These two writes must not be combined.

Step 5: System Power Management driver asserts reset to XUSB then disables its clocks.

- Set the following CAR register bits to '1' to assert reset to XUSB

  CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_HOST_RST] for host mode

  CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_DEV_RST] for device mode

  CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST] for SuperSpeed ports

- Set the following CAR register bits to '1' to disable the clocks to individual XUSB partitions.

  CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0[CLR_CLK_ENB_XUSB_HOST] for host mode

  CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0[CLR_CLK_ENB_XUSB_DEV] for device mode

  CLK_RST_CONTROLLER_CLK_ENB_W_CLR_0[CLR_CLK_ENB_XUSB_SS] for SS ports

Step 6: System Power Management driver disables the XUSB power rails.

- Program the following PMC register bits to disable the power rail to XUSB host

  APBDEV_PMC_PWRGATE_TOGGLE_0[START] to 'disable'

  APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to 'XUSBB'

- Program the following PMC register bits to disable the power rail to XUSB device

  APBDEV_PMC_PWRGATE_TOGGLE_0[START] to 'disable'

  APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to 'XUSBC'

- Program the following PMC register bits to disable the power rail to XUSB SuperSpeed

  APBDEV_PMC_PWRGATE_TOGGLE_0[START] to 'disable'

  APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to 'XUSBA'

> **Note:** Only one partition can be power gated at a time.

Step 7: xHCI PEP driver and XUSB device mode driver initiate the signal sequence to enable the XUSB SS wake detection logic for the SuperSpeed ports assigned to host and device.

- Write ‘1’ to the following XUSB PADCTL register bit to assert the vcore_off signal.

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_VCORE_DOWN]

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_VCORE_DOWN]

23.7.5.2 SuperSpeed Partition ELPG Entry

Step 1: xHCI FW and XUSB device mode driver perform SuperSpeed port specific context save operation.

Step 2: xHCI PEP driver and XUSB device mode driver enable the XUSB wakeup interrupts for the SuperSpeed and USB2.0 ports assigned to host and device.

Step 3: xHCI PEP driver and XUSB device mode driver initiate the signal sequence to enable the XUSB SSwake detection logic for the SuperSpeed ports assigned to host and device.

Step 4: System Power Management driver asserts reset to XUSB SuperSpeed partition then disables its clocks.

- Set the following CAR register bits to ‘1’ to assert reset to XUSB

  CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST] for SuperSpeed ports

- Set the following CAR register bits to ‘1’ to disable the clocks to individual XUSB partitions.

  CLK_RST_CONTROLLER_CLK_ENB_W_CLR_0[CLR_CLK_ENB_XUSB_SS] for SS ports

Step 5: System Power Management driver disables the XUSB SuperSpeed partition power rails.

- Programming the following PMC register bits to disable the power rail to XUSB SuperSpeed

  APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘disable’

  APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBA’

> **Note:** Only one partition can be power gated at a time.

Step 6: xHCI PEP driver and XUSB device mode driver initiate the signal sequence to enable the *XUSB* SS wake detection logic for the SuperSpeed ports assigned to host and device.

23.7.5.3 Device Mode Partition ELPG Entry

Step 1: XUSB device mode driver performs XUSB_DEVICE specific context save operations. XUSB device mode driver performs XUSB IPFS specific context save operation.

Step 2: System Power Management driver programs the PMC USB2.0 sleepwalk logic for ports assigned to XUSB device mode according to the PMC USB Wakeup Document.

System Power Management driver should enable the wake events according to the port states and Wake-on-Connect, Wake-on-Disconnect, and Wake-on-Over-Current settings of the USB2 ports accordingly.

> The line wakeup event from PMC is shared between XUSB and USB2 controller. There is no per controller masking for the line wakeup event, and when the sleepwalk logic triggers the event, both XUSB and USB2 will log the status even when they are not power gated or clock gated. Thus the system power management driver should ensure the wake status is cleared before enabling the sleepwalk logic.

Step 3: XUSB device mode driver enable the XUSB wakeup interrupts for the SuperSpeed and USB2.0 ports assigned to device as described in step 3 of section on “.All Partitions ELPG Entry” above.

Step 4: XUSB device mode driver initiate the signal sequence to enable the XUSB SSwake detection logic for the SuperSpeed ports assigned device as described in step 4 of section on “.All Partitions ELPG Entry” above..

Step 5: System Power Management driver asserts reset to XUSB then disables its clocks.

- Set the following CAR register bits to ‘1’ to assert reset to XUSB

  CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_DEV_RST] for device mode

- Set the following CAR register bits to ‘1’ to disable the clocks to individual XUSB partitions.

  CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0[CLR_CLK_ENB_XUSB_DEV] for device mode

Step 6: System Power Management driver disables the XUSB power rails.

- Programming the following PMC register bits to disable the power rail to XUSB device

  APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘disable’

  APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBC’

> **Note:** Only one partition can be power gated at a time.

Step 7: XUSB device mode driver initiate the signal sequence to enable the XUSB SS wake detection logic for the SuperSpeed ports assigned to device as described in step 7 of section on “.All Partitions ELPG Entry” above..

- Write ‘1’ to the following XUSB PADCTL register bit to assert the vcore_off signal.

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_VCORE_DOWN]

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_VCORE_DOWN]

23.7.5.4 All Partitions ELPG Exit

When the wake event is not generated by SuperSpeed ports, the SuperSpeed related programming should be skipped, where the SuperSpeed partition should exit ELPG due to wake events detected by SuperSpeed ports.

Step 1:

- System Power Management driver enables the XUSB power rails. Programming the following PMC register bits to enable the power rail to XUSB host

  APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’

  APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBB’

- Programming the following PMC register bits to disable the power rail to XUSB device

  APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’

  APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBC’

- Programming the following PMC register bits to disable the power rail to XUSB SuperSpeed

  APBDEV_PMC_PWRGATE_TOGGLE_0[START] to ‘enable’

  APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to ‘XUSBA’

> **Note:** Only one partition can be power gated at a time.

Step 2: System Power Management driver enables XUSB clocks.

- Set the following CAR register bits to ‘1’ to enable the clocks to individual XUSB partitions.

  CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_XUSB_HOST] for host mode

  CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_XUSB_DEV] for device mode

  CLK_RST_CONTROLLER_CLK_ENB_W_SET_0[SET_CLK_ENB_XUSB_SS] for SS ports

Step 3: System Power Management driver deasserts reset to XUSB.

- Set the following CAR register bits to '0' to deassert reset to XUSB:

  CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_HOST_RST] for host mode

  CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_DEV_RST] for device mode

  CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST] for SuperSpeed ports

- Wait 1 µs

Step 4: xHCI PEP driver and XUSB device mode driver disables the XUSB SS wakeup interrupts.

- Set the following XUSB PADCTL register bits to '0' to disable the interrupts:

  XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT0_WAKE_INTERRUPT_ENABLE]

  XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT1_WAKE_INTERRUPT_ENABLE]

  XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT0_WAKE_INTERRUPT_ENABLE]

  XUSB_PADCTL_ELPG_PROGRAM_0[USB2_PORT1_WAKE_INTERRUPT_ENABLE]

  XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT0_WAKE_INTERRUPT_ENABLE]

  XUSB_PADCTL_ELPG_PROGRAM_0[USB2_HSIC_PORT1_WAKE_INTERRUPT_ENABLE]

Step 5: xHCI PEP driver and XUSB device mode driver initiate the signal sequence to disable the XUSB SS wake detection logic.

- Write '0' to the following XUSB PADCTL register bits to deassert the vcore_off signal:

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_VCORE_DOWN]

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_VCORE_DOWN]

- Write '0' to the following XUSB PADCTL register bits to deassert the clamp_en and clamp_en_early signals:

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN]

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_CLAMP_EN]

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP0_ELPG_CLAMP_EN_EARLY]

  XUSB_PADCTL_ELPG_PROGRAM_0[SSP1_ELPG_CLAMP_EN_EARLY]

> **Note:** The write to clear vcore_off cannot be combined with the write to clear clamp_en and clamp_en_early.

Step 6: System Power Management driver programs the PMC USB2.0 sleepwalk logic for ports assigned to XUSB host mode and XUSB device mode to disable the sleepwalk logic.

Step 7: xHCI PEP driver and XUSB device mode driver perform XUSB IPFS specific context restore operation for registers listed in Step 1 of section on ".All Partitions ELPG Entry" above, then enables XUSB IPFS operation.

- Set the following XUSB IPFS register bit to '1':

  XUSB_{HOST,DEV}_CONFIGURATION_0[EN_FPCI]

xHCI PEP driver performs XUSB specific context restore operation. XUSB device mode driver performs XUSB_DEVICE specific context restore operation.

Step 8: xHCI PEP Driver loads XUSB firmware. Refer to the FW Boot Loader section. xHCI PEP Driver notifies XUSB firmware whether context of SuperSpeed Partition should be restored.

Step 9: xHCI driver performs context restore operation.

23.7.5.5 SuperSpeed Partition ELPG Exit

Step 1: System Power Management driver enables the XUSB power rails.

- Programming the following PMC register bits to disable the power rail to XUSB SuperSpeed

  APBDEV_PMC_PWRGATE_TOGGLE_0[START] to 'enable'

  APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to 'XUSBA'

> **Note:** Only one partition can be un-power gated at a time.

Step 2: System Power Management driver enables XUSB clocks.

- Set the following CAR register bits to '1' to enable the clocks to individual XUSB partitions.

  CLK_RST_CONTROLLER_CLK_ENB_W_SET_0[SET_CLK_ENB_XUSB_SS] for SS ports

Step 3: xHCI PEP driver and XUSB device mode driver initiate the signal sequence to disable the XUSB SS wake detection logic for the SuperSpeed ports assigned to host and device as described in Step 3 of section on "All Partitions ELPG Exit" above.

Step 4: xHCI PEP driver and XUSB device mode driver disable the XUSB SS wakeup interrupts.

- Set the following XUSB PADCTL register bits to '0' to disable the interrupts.

  XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT0_WAKE_INTERRUPT_ENABLE]

  XUSB_PADCTL_ELPG_PROGRAM_0[SS_PORT1_WAKE_INTERRUPT_ENABLE]

Step 5: System Power Management driver deasserts reset to XUSB.

- Set the following *CAR* register bits to '0' to deassert reset to XUSB

  CLK_RST_CONTROLLER_RST_DEVICES_W_0[SWR_XUSB_SS_RST] for SuperSpeed ports

Step 6: xHCI FW and XUSB device mode driver performs their SuperSpeed port specific context restore operations.

23.7.5.6 Device Mode Partition ELPG Exit

Step 1: System Power Management driver enables the XUSB power rails.

- Programming the following PMC register bits to disable the power rail to XUSB device

  APBDEV_PMC_PWRGATE_TOGGLE_0[START] to 'enable'

  APBDEV_PMC_PWRGATE_TOGGLE_0[PARTID] to 'XUSBC'

> **Note:** Only one partition can be un-power gated at a time.

Step 2: System Power Management driver enables XUSB clocks.

- Set the following CAR register bits to '1' to enable the clocks to individual XUSB partitions.

  CLK_RST_CONTROLLER_CLK_ENB_U_SET_0[SET_CLK_ENB_XUSB_DEV] for device mode

Step 3: XUSB device mode driver initiate the signal sequence to disable the XUSB SS wake detection logic for the SuperSpeed ports assigned to device as described in Step 3 of section on "All Partitions ELPG Exit" above.

Step 4: XUSB device mode driver disables the XUSB SS wakeup interrupts for the ports assigned to device as described in Step 4 of section on "All Partitions ELPG Exit" above.

Step 5: System Power Management driver deasserts reset to XUSB.

- Set the following CAR register bits to '0' to deassert reset to XUSB

  CLK_RST_CONTROLLER_RST_DEVICES_U_0[SWR_XUSB_DEV_RST] for device mode

Step 6: System Power Management driver programs the PMC USB2.0 sleepwalk logic for ports assigned to XUSB device mode to disable the sleepwalk logic.

Step 7: XUSB device mode driver perform XUSB IPFS specific context restore operation for registers listed in Step 1 of the section on "All Partitions ELPG Entry" above, then enables XUSB IPFS operation.

- Set the following XUSB IPFS register bits to '1'

  XUSB_DEV_CONFIGURATION_0[EN_FPCI]

XUSB device mode driver performs XUSB_DEVICE specific context restore operation.

23.7.6 Initialization and Shutdown Procedure for USB

1. Bring up USB3 clocks by writing 1 to CLK_ENB_USB3 in ARCLK_RST_CLK_OUT_ENB_H register.
2. Set and clear the SWR_USB3_RST in RST_DEVICES_H register to bring the USB block out of reset.
3. Set the UTMIP_RESET bit in USB_SUSP_CTRL register to 1 to keep UTMIP3 PHY in reset.
4. Program UTMIP and PLLU parameters.
   - Set register UTMIP_MISC_CFG0 UTMIP_SUSPEND_EXIT_ON_EDGE (bit 22) to 0.
   - Set the PLLU_ENABLE to 1, and never ever de-assert it.
5. Set UTMIP_RESET bit in USB_SUSP_CTRL register to 0 to bring UTMIP3 out of reset.
   - Wait until PHY_CLKVALID is set to 1.
6. Set USB controller and PHY to suspend by writing 1 to PHCD in USB2D_PORTSC1 register. This will reduce the power consumption on USB3 controller and UTMIP3 PHY.
   - Wait until PHY_CLKVALID is set to 0.
7. Set all PD bits required to bring USB pads to low power mode.
   - UTMIP_OTGPD in UTMIP_BIAS_CFG0 is needed for VBUS detection, and so it should be set to 0.
   - If using VBUS_WAKEUP for VBUS detection, then UTMIP_BIASPD in UTMIP_BIAS_CFG0 can be set to 1 to save power. But if using A_SESS_VLD or any other VBUS sensor for VBUS detection, then this bit should be set to 0.
   - ID_PU in USB_PHY_VBUS_WAKEUP_ID should be set to 1 to enable ID detection.
   - UTMIP_IDPD_SEL in UTMIP_BIAS_CFG0 should be set to 0.
8. To interrupt the processor on VBUS or ID detection, software can set the following:
   - Enable appropriate VBUS interrupt enable, for example, I using A_SESS_VLD sensor for VBUS detection, set A_SESS_VLD_INT_EN in USB_PHY_VBUS_SENSORS register to 1.
   - Enable ID detection interrupt by setting ID_INT_EN in USB_PHY_VBUS_WAKEUP_ID register.
9. When (VBUS=1) or (ID=0) is detected, do the following:
   - Bring USB3 controller and PHY out of suspend by pulsing SUSP_CLR in USB_SUSP_CTRL register by first writing 1 and then writing 0.

10. Wait until PHY_CLKVALID is set to 1.
    - Clear the PD bits that are required for the normal operation.
11. From this point on, the driver can run the normal operations.
    - UTMIP should only be suspended by writing 1 to PHCD in USB2D_PORTSC1 register when USB cable is connected and only after USB bus is in suspend, for both device and host modes.
12. If a cable disconnect is detected (VBUS = 0) or (ID = 1), then go back to step 4.

23.7.7 Recommended PHY Settings

Contact your local NVIDIA representative for PHY settings specific to your design.

23.7.8 BIAS PAD Configuration

UTMIP BIAS pad is shared across USB_OTG and USB3 controllers and the below common bias pad settings need to be configured from USB1 controller only.

USB1_UTMIP_BIAS_CFG0_0:

- UTMIP_BIASPD
- UTMIP_HSCHIRP_LEVEL
- UTMIP_HSSQUELCH_LEVEL
- UTMIP_HSDISCON_LEVEL_MSB
- UTMIP_HSDISCON_LEVEL
- UTMIP_ACTIVE_TERM_OFFSET
- UTMIP_ACTIVE_PULLUP_OFFSET
- UTMIP_VBUS_LEVEL_LEVEL
- UTMIP_SESS_LEVEL_LEVEL

USB1_UTMIP_BIAS_CFG1_0:

- UTMIP_FORCE_PDTRK_POWERUP
- UTMIP_FORCE_PDTRK_POWERDOWN

Following is the sequence to configure above parameters of the BIAS pad.

1. Enable clock to USB1 controller
2. Set any of above parameters as required.
3. Disable clock to USB1 controller if it is not used.

23.7.9 BOOT ROM Initialization Sequence for USB Recovery

1. Program PLL_U.
    - Set the PLLU_BASE register fields, PLLU_DIVM, PLLU_DIVN, PLLU_VCO_FREQ, PLLU_BYPASS and PLLU_ENABLE fields as described in this document.
2. Configure USB
    a. Bring up USB1 (USB_OTG) clocks by writing 1 to CLK_ENB_USBD in CLK_OUT_ENB_L register.

b. Assert and deassert master USBD reset in CAR (SWR_USBD_RST in RST_DEVICES_L register) to bring USB1 out of reset.

c. Stop crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN in UTMIP_MISC_CFG1 register to 0. This only stops the crystal clocks in the UTMIP units.

   Default value of the USB1_UTMIP_PHY_XTAL_CLOCKEN is 1, now changes to 0.

d. Programming automatic PLL start times.

   Setting USB1_IF_UTMIP_PLLU_ENABLE_DLY_COUNT, UTMIP_PLLU_STABLE_COUNT, UTMIP_PLL_ACTIVE_DLY_COUNT and UTMIP_XTAL_FREQ_COUNT as per the values given in the document.

e. Programming the tracking duration.

   Setting USB1_IF_UTMIP_BIAS_CFG1.UTMIP_BIAS_PDTRK_COUNT as per the values given in the document.

f. Program the debouncer length times.

   Setting UTMIP_DEBOUNCE_CFG0.UTMIP_BIAS_DEBOUNCE_A field.

g. Program various static parameters of the USB UTMIP1.

   Set UTMIP_TX_CFG0.UTMIP_FS_PREAMBLE_J to 0x1.

   Set UTMIP_BAT_CHRG_CFG0.UTMIP_PD_CHRG to 1.

   Set UTMIP_XCVR_CFG0.UTMIP_XCVR_LSBIAS_SEL to 0.

   Set third bit of UTMIP_SPARE_CFG0 to 1. i.e UTMIP_SPARE_CFG0 [3] to 1.

   Set UTMIP_HSRX_CFG0. UTMIP_IDLE_WAIT as per the values given in this document.

   Set UTMIP_HSRX_CFG0.UTMIP_ELASTIC_LIMIT to 16.

   Set UTMIP_HSRX_CFG1.UTMIP_HS_SYNC_START_DLY to 9

h. Restart the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN in UTMIP_MISC_CFG1 register to 1 for USB.

3. Wait for cable connect on USB UTMIP1 port. When cable is connected on USB UTMIP1 port, continue to the next step.
4. Bring UTMIP1 out of reset by writing 0 to UTMIP_RESET bit of USB1_IF_SUSP_CTRL register.
5. Wait until USB1_IF_USB_SUSP_CTRL.PHY_CLK_VALID is set to 1.
6. Then perform USB controller initialization.
   a. Reset the bus.
   b. Wait until the bus comes out of reset.
   c. Set the controller in device mode.
   d. Perform USB operations.

23.7.10 Performance Settings for USB Controllers

To meet USB's strict bandwidth/latency requirements, some AHB programming needs to be done. The following gives a guideline on the programming requirements to achieve maximum performance from USB:

- The burst size for USB controller should be programmed to 8 in the register USB2D_BURSTSIZE. Both TXPBURST and RXPBURST fields should be programmed to the same value of 8.
- The field ENB_FAST_REARBITRATE for AHB_MEM gizmo should be set to 1 in the register AHB_GIZMO_AHB_MEM.

- The field IMMEDIATE for USB gizmos should be set to 1 in the register AHB_GIZMO_USB, AHB_GIZMO_USB2 or AHB_GIZMO_USB3 depending on the controller in use.
- USB controllers should be set as high-priority masters on AHB by setting the bits corresponding to each USB controller to 1 in AHB_PRIORITY_SELECT field in the register AHB_ARBITRATION_PRIORITY_CTRL and setting the priority weight to 7 by setting the field AHB_PRIORITY_WEIGHT in the same register. USB master numbers are 6 for USB1, 18 for USB2 and 17 for USB3. The priority weight could be relaxed depending on requirements from other AHB masters in the system as required for different use cases.
- Prefetch engine needs to be setup correctly to enable prefetching of transmit data packets for USB masters. Each USB master needs one channel on the prefetch engine. There are 4 channels on the prefetch engine. If all 3 USB masters enable one channel at the same time, it would leave one more channel for another AHB master. Each prefetch channel is controlled by the register AHB_AHB_MEM_PREFETCH_CFG[NO] where NO=1,2,3,4. To enable prefetch for a USB controller on a channel, program AHB_MST_ID_USB, AHB_MST_ID_USB2 or AHB_MST_ID_USB3 in the field AHB_MST_ID for the register AHB_AHB_MEM_PREFETCH_CFG[NO]. The field ADDR_BNDRY should be set to log2 (buffer size) according to the buffer size required for the corresponding USB master. The field SPEC_THROTTLE should be set to 0, and the field INACTIVITY_TIMEOUT should be set to 0x800.
- When a particular USB controller is in Host mode, the field TXFIFOTHRES in the register USB2D_TXFILLTUNING (offset 0x154) should be set to 0x10.

All this programming need to be done before setting the RS bit in USB2D_USBCMD to RUN (1) for the corresponding USB controller.

23.7.11 USB Controller Handling of USB Resume Sequence

Following are the steps for how the USB controller handles a USB resume while the port is under PMC control:

1. PMC maintains suspend on the bus.
2. Auto resume happens, system starts restoring USB controller.
3. Set RUN bit from USB controller to start SOFs.
4. USB controller brought back to suspend state.
5. Switch USB bus from PMC to USB controller.
6. Wait until resume complete notified from USB controller. This is handled by the EHCI driver.
7. Further USB communication can start from here.

23.8 UTMIP Programming Guidelines

A Tegra 4 device has a total of 2 UTMI+ interfaces, where the second controller does not have a UTMI+ interface. This results in two cabled USB ports. These ports are not part of a multiport host configuration. They are completely independent ports and can be configured in any combination of device or host. Internally, these ports correspond to USB controllers 1 and 3, where controller 1 is often implicitly numbered. Externally, these are USB ports 0 and 1. All UTMI+ PHY interfaces are identical instances of the same design. All three controllers use latest edition of USB core controller.

23.8.1 Holding USB in Reset

It is important that most static configuration of the UTMIP (and USB) be done while the unit is held in reset. For example, holding reset ensures that PLL_U is disabled and can be reconfigured. It also ensures that transactions are not occurring on the USB interface. Holding reset in both a controller and its corresponding PHY can be achieved by using the RST bit of the USB2D_USBCMD register.

23.8.2 PLL_U Programming

The USB ports use a cascaded PLL scheme to guarantee a high clock quality. First there is a PLL_U, which is the source clock for both the USB PLLs that are dedicated to ports. PLL_U (of type CLKPLL960_USB) produces reference clocks for all forms of USB (HSIC USB, UTMIP, UHSIC). Table 60 describes the parameter settings that are dependent on the crystal clock reference frequency. The parameters are specified in decimal notation.

Table 60: 480 MHz PLL_U Output Frequency Fo = (Fi·N)÷(M· 2) With VCO_FREQ=0

Fi (MHz)	12.0MHz	13.0MHz	16.8MHz	19.2MHz	26.0MHz	38.4 MHz	48.0MHz
N (PLLU_DIVN)	960	960	400	200	960	200	960
M (PLLU_DIVM)	12	13	7	4	26	4	12

These parameters can be found in the PLLU_BASE register. The PLLU_OVERRIDE bit should be set to 0. PLLU_VCO_FREQ parameter must also be set to 0.

Note: PLL_U must be properly configured in the Boot ROM. DO NOT change the parameters of PLL_U while the unit is running. The same applies to the USB_PHY_PLL.

23.8.3 PLL_U and USB_PHY_PLL Automatic Startup Times

The PLL_U and USB_PHY_PLL automatic startup times are used in reset and suspend modes to kick start the PLLs. Software should not manually force the PLL up and down because it cannot do so rapidly enough to meet the protocol. The following table lists the start times, which must be set up in the Boot ROM.

Table 61: PLL Automated Start Times (Must be Set Up in the Boot ROM)

Crystal Frequency	12.0 MHz	13.0 MHz	16.8 MHZ	19.2 MHz	26.0 MHz	38.4 MHz	48.0 MHz
PLLU_ENABLE_DELAY_COUNT	2	2	3	3	4	5	6
PLLU_STABLE_COUNT	47	51	66	75	102	150	188
PLL_ACTIVE_DLY_COUNT	8	9	11	12	9	24	31
XTAL_FREQ_COUNT	118	127	165	188	254	375	469

PLLU_ENABLE_DLY_COUNT should be set for at least 1 microsecond, PLLU_STABLE_COUNT should be set for at least 1 ms, the PLL_ACTIVE_DLY_COUNT should be at least 10 ms, and the XTAL_FREQ_COUNT set for about 2.5 ms.

23.8.4 Fuse Programming

The Tegra 4 device incorporates fuses to account for process variation in high speed data eye swing. The high speed data eye is controlled through the SETUP[6:0] transceiver pad parameter of the USB control registers. To set this parameter via the fuses, set the UTMIP_SPARE_CFG0[3] bit on all USB ports. This must be done in the Boot ROM. The default is to maintain backward compatibility; which implies no fuses are used by default. The objective is to get the data eye swing as close as possible to 400mV without falling below the mark under typical operating conditions.

23.8.5 Programming the Tracking Length Time

The PD_TRK signal is used to power down the bias cell. After 25 microseconds of bias cell operation, the PD_TRK signal can be turned high to save power. This can be automated by programming a timing interval as given in the following table.

Table 62: 25 µs Timer on Bias Cell Tracking (for Set Up in the Boot ROM)

Crystal Frequency	12.0 MHz	13.0 MHz	16.8 MHZ	19.2 MHz	26.0 MHz	38.4 MHz	48.0 MHz
UTMIP_BIAS_PAD_TRK_COUNT (UTMIP_BIAS_CFG1[7:3])	5	6	7	8	11	15	19

All values in the table are decimal. This parameter must be set in the Boot ROM and while the crystal clock is stopped. To stop the crystal clock, lower the UTMIP_PHY_XTAL_CLOCKEN bit of register UTMIP_MISC_CFG1 and then re-enable it when done.

Because there is more than one UTMIP sharing the same bias cell, power down is done through a daisy chain, requiring power down across all ports.

23.8.6 Powering Down a USB Port Outside of Deep Power Down (DPD)

When a port is known to be disabled, there needs to be a mechanism to stop its power consumption for situations outside of DPD. This section addresses disabling a port while powered up.

Powering down of USB at times other than deep power down should be done by setting all specialized PLL and pad power down pins instead of using the global E_DPD pins of the USB analog components. It is safer to power down the cells through the traditional power down pins when the 3.3V and 1.2V supplies might still be on. This is achieved by setting the following pad controls from UTMIP register space. Before setting the power down bits; save previous registers values so that they may be restored. This type of power down should not be done in the Boot ROM.

Table 63: UTMIP Pad Controls

Pad Control	Analog Cell	Register Bit Settings
PD	Transceiver	UTMIP_FORCE_PD_POWER_DOWN=1
PD2	Transceiver	UTMIP_FORCE_PD2_POWER_DOWN=1
PD_ZI	Transceiver	UTMIP_FORCE_PDZI_POWER_DOWN=1
PD_DR	Transceiver	UTMIP_FORCE_PDDR_POWER_DOWN=1
PD_CHRP	Transceiver	UTMIP_FORCE_PDCHRP_POWER_DOWN=1
PD_DISC	Transceiver	UTMIP_FORCE_PDDISC_POWER_DOWN=1
PD_CHG	Transceiver	UTMIP_PD_CHG=1
PD	Bias	UTMIP_BIASPD=1
PD_TRK	Bias	UTMIP_FORCE_PDTRK_POWER_DOWN=1
OTG_PD	Bias	UTMIP_OTGPD=1
ID_PD	Bias	UTMIP_IDPD_SEL=1, UTMIP_IDPD_VAL=1
VBUS_WAKEUP_PD	Bias	UTMIP_VBUS_WAKEUP_POWER_DOWN=1
ENABLE	PHY PLL	UTMIP_FORCE_PLL_ENABLE_POWERDOWN=1
ACTIVE	PHY PLL	UTMIP_FORCE_PLL_ACTIVE_POWERDOWN=1
PD_SAMP_A/C	PHY_PLL	UTMIP_FORCE_PD_SAMP_A/C_POWERDOWN=1
ENABLE	PLL U	UTMIP_FORCE_PLLU_POWERDOWN=1
UTMIP	PHY	UTMIP_PHY_XTAL_CLOCKEN=0

23.8.7 Extending Debouncer Period Length

USB debouncers provide a programmable debounce to alleviate the software burden. This is done with the UTMIP_DEBOUNCE_TIME_SCALE parameter located at UTMIP_BIAS_CFG1[13:8]. The default implies a timescale factor of 1. The timescale should not be programmed at Boot ROM time. The timescale register field incremented by 1 multiplies the debounce duration for all debouncers.

In the Boot ROM, the A debounce period should be set to 10 ms. This is dependent on the crystal frequency scale. The following table shows how it is programmed.

Table 64: 10 ms Timer on the A Debounce Period

Crystal Frequency	12.0 MHz	13.0 MHz	16.8 MHZ	19.2 MHz	26.0 MHz	38.4 MHz	48.0 MHz
UTMIP_BIAS_DEBOUNCE_A	30000	32500	42000	48000	65000	96000	120000

To program values of 96000 and 120000 for crystal clock frequencies of 38.4 and 48 MHz, respectively, these steps must be followed:

1. In the debounce register, program 48000 for a 38.4 MHz crystal clock or 60000 for a 60 MHz clock.
2. Set UTMIP_BIAS_DEBOUNCE_TIMESCALE to 1 by writing into the USB2_UTMIP_BIAS_CFG1[1] register field.

23.8.8 Deep Power Down Behavior

Tegra 4 devices' deep power down behavior for USB wake-up events is controlled in the PMC via registers USB_DEBOUNCE_DLY, USB_AO and the WAKE_MASK (USB_EVENT field) of the PMC unit. These control the wake-up events, their debounce duration, and their debounce length while powered down. USB port 0 has a possible event on ID or VBUS detection. These can be configured at startup time. These registers take over from the UTMIP registers when the chip is powered down. The programming depends on the context of the chip. For example, if USB port 0 is a dedicated host port then the VBUS wake-up event should be kept powered down. Conversely, a dedicated device mode port may not want to wake-up on the detection of an ID connector and can chose to power down ID.

23.8.9 PHY Resets

Additional controls have been provided to stop the USB PHY (and its clocks) by issuing a reset to the PHY only (as opposed to a reset encompassing the controller and the PHY). This is akin to the Reset found in the UTMI+ standard. This is done in both controllers. See the UTMIP_RESET field of the USB_SUSP_CTRL register for more information.

23.8.10 Static Boot ROM Configuration for UTMIP

The boot ROM configuration for UTMIP should be done while the unit is held in Reset. For correct behavior, the following sequence MUST be followed:

1. Apply Reset to the USB controller (and UTMIP).
2. Run the crystal clock for approximately 5 microseconds while the UTMIP is in reset.
3. Stop the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN low. This only stops the crystal clocks in the UTMIP units.
4. Program PLL_U as described in the PLL_U programming section (section 3).
5. Program automatic PLL start times as described in the "PLL_U and USB_PHY_PLL Automatic Startup Times."
6. Remove power downs from USB_PHY_PLL ACTIVE/ENABLE/PD_SAMP_A/C and PLL_U.
7. Program the tracking duration as described in "Programming the Tracking Length Time." Remove the power downs from PD (Bias) and disable PD_TRK. After 25 µs (tracking time), enable PD_TRK by writing 0 into UTMIP_FORCE_PDTRK_POWER_DOWN.
8. Once PD (Bias) is disabled, disable OTG_PD after 1 µs.
9. Remove powerdowns from ID_PD and VBUS_WAKEUP_PD. This can be removed earlier, as there is no timing or sequential relation with other signals.
10. Once the PLL_U, USB_PHY_PLL, and Bias pad are ready, remove PD (Transceiver). After 400 ns, the USB pad is ready for HS/FS/LS operation.
11. There are other powerdowns which can be removed if needed (as such not needed for Boot ROM). These are PD / PD_ZI / PD_DR / PD_CHRP / PD_DISC / PD_CHG.

12. Program the debouncer length time as described in ”Extending Debouncer Period Length.”
13. Program various static parameters of the UTMIP as described in ”Miscellaneous Boot ROM Fields.”
14. Restart the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN.
15. Resume any previous USB programming work that falls outside of UTMIP and release reset.From the UTMIP perspective it does not matter when reset is released, as long as it is after step 8. It will take about 3.5 ms before the 60 MHz clock appears once reset is released.

From a UTMIP perspective, ensure that all the cabled USB port PHYs are configured similarly in the Boot ROM.

23.8.11 EHCI Deviations

- Embedded Transaction Translator – Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Embedded design interface – This core does not have a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.

Programmable Physical Interface Behavior

This design supports multiple Physical interfaces which can operate in differing modes when the core is configured with software programmable Physical Interface Modes. Software programmability allows the selection of the Physical interface part during the board design phase instead of during the chip design phase. The control bits for selecting the Physical Interface operating mode have been added to the HOSTPCx register providing a capability that is not defined by EHCI.

Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the PORTSCx register for a minimum duration of 10ms. Due to the complexity required to support the attachment of devices that are not high speed, there are counter already present in the design that can count the 10ms reset pulse to alleviate the requirement of the software to measure this duration. The basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a ‘1’ to the reset the device.
- Software shall write a ‘0’ to the reset the device after 10 ms.
- Driver needs to wait until port change interrupt is asserted and port reset is cleared

> **Note:** Should the EHCI host controller driver attempt to write a ‘0’ to the reset bit while a reset is in progress the write will be ignored and the reset will continue until completion.

- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device is now operational and at this point the port speed has been determined.

Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices. Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.

- A 2-bit Port Speed indicator (PSPD) has been added to HOSTPCx to provide the current operating speed of the port to the host controller driver.

Port Reset duration is 55 ms instead of 50 ms.

Port Suspend (PORTSC.Suspend) bit is set immediately after setting the bit. As per EHCI a 4 ms delay is expected for this bit to be set.

Tegra 4 devices also have the following cases that are not directly compliant to EHCI:

- USB controller/PHY interface initialization
- PHY Clock shutdown/wakeup procedure during suspend/resume/connect/disconnect
- For HSIC, connect process and bus reset process is not EHCI compliant – particularly no interrupts for connect.
- PMC logic use for special low power modes during suspend/LP0

23.9 USB Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

23.9.1 USB1 Controller Registers

23.9.1.1 USB2_CONTROLLER_USB2D_ID_0

USB2D Identification Register

Offset: 0x0 | Read/Write: RO |Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	CIVERSION: Identifies the CI version
28:25	X	VERSION: Identifies the version of the core
24:21	X	REVISION: Revision number of the USB controller. This is set to 0x0.
20:16	X	TAG: Identifies the tag of the core
15:8	X	NID: One’s complement version of ID. This field is set to 0xF9.
7:0	X	ID: Configuration number. This field is set to 0x06

23.9.1.2 USB2_CONTROLLER_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 0x8 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
3:1	X	NPORT: VUSB_HS_NUM_PORT-1: This host controller has only 1 port. So this field will always be 0.
0	X	HC: VUSB_HS_HOST: Indicates support for host mode. Set to 1.

23.9.1.3 USB2_CONTROLLER_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

Offset: 0xc | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: Number of endpoints supported by this device controller. Set to 16. This includes control endpoint 0.
0	X	DC: Device capable: Set to 1 indicating support for device mode.

23.9.1.4 USB2_CONTROLLER_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 0x10 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total number of address bits for the transmit buffer of each transmit endpoint. Set to 7. Each transmit buffer is 128 words deep.
15:8	X	TXADD: VUSB_HS_TX_ADD: Total number of address bits for the transmit buffer. Set to 11. The total depth of the transmit buffer is 2048 words.
7:0	X	TCBURST: VUSB_HS_TX_BURST: Maximum burst size supported by the transmit endpoints for data transfers. Set to 8.

23.9.1.5 USB2_CONTROLLER_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 0x14 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total number of address bits for the receive buffer. Set to 7. The total depth of the receive buffer is 128 words
7:0	X	RXBURST: VUSB_HS_RX_BURST: Maximum burst size supported by the receive endpoints for data transfers. Set to 8.

23.9.1.6 USB2_CONTROLLER_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

Offset: 0x100 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
7:0	X	CAPLENGTH: Indicates which offset to add to the register base address at the beginning of the Operational Register. Set to 0x40.

23.9.1.7 USB2_CONTROLLER_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 0x102 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	HCIVERSION: Contains a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. This host controller supports EHCI revision 1.00.

23.9.1.8 USB2_CONTROLLER_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

Offset: 0x104 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
27:24	X	N_TT: Number of Transaction Translators: Indicates the number of embedded transaction translators associated with the USB2.0 host controller. This field is always set to 1 indicating only 1 embedded TT is implemented in this implementation. This is a non-EHCI field to support embedded TT.
23:20	X	N_PTT: Number of Ports per Transaction Translator: Indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. Field always equals N_PORTS. This is a non-EHCI field to support embedded TT.
15:12	X	N_CC: Number of Companion Controller: Indicates the number of companion controllers. This field is set to 0.
11:8	X	N_PCC: Number of Ports per Companion Controller: Indicates the number of ports supported per internal companion controller. This field is set to 0.
4	X	PPC: Port Power Control: Indicates whether the host controller implementation includes port power control. 1 = Ports have port power switches 0= Ports do not have port power switches. This field affects the functionality of the port Power field in each port status and control register. This field is set to 1.
3:0	X	N_PORTS: Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. This field is fixed to 1, since this host controller only supports 1 port.

23.9.1.9 USB2_CONTROLLER_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

Offset: 0x108 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18	X	PPC: Per-Port Change Event Capability - RO. Default = 1b. This field indicates the support for per-port change events. This field is related to the USBCMD PPE field, USBSTS PPCI field, and USBINTR PPCE field.
17	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to USBCMD HIRD field, POSTSCx SSTS and DA fields and HOSTPCx LEN, BA, and EPLPM fields.
15:8	X	EECP: EHCI Extended Capabilities Pointer: Indicates a capabilities list exists. A value of 00h indicates no extended capabilities are implemented. For this implementation this field is always "0".
7:4	X	IST: Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be "0".

Bit	Reset	Description
2	X	ASP: Asynchronous Schedule Park Capability. 1 = (Default) the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register. This field is always 1.
1	X	PFL: Programmable Frame List Flag. 0 = System software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. 1 = System software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This field will always be "1".

23.9.1.10 USB2_CONTROLLER_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

Offset: 0x120 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	DCIVERSION: The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

23.9.1.11 USB2_CONTROLLER_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 0x124 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
8	X	HC: Host Capable: 1 = This controller can operate as an EHCI compatible USB 2 0 host controller. This field is set to 1.
7	X	DC: Device Capable: 1 = Controller can operate as a USB 2.0 device. This field is set to 1.
5	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the DEVLCx ASUS, STL, BA, and NYT fields.
4:0	X	DEN: Device Endpoint Number: Number of endpoints built into the device controller. This is set to 16.

23.9.1.12 USB2_CONTROLLER_USB2D_EXTSTS_0

USB2D EXTSTS Register

Offset: 0x128 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000x

Bit	R/W	Reset	Description
4	RW	0x0	TI1: General Purpose Timer Interrupt 1 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
3	RW	0x0	TI0: General Purpose Timer Interrupt 0 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
2	RW	0x0	UPA: USB Host Periodic Interrupt (USBHSTPERINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
1	RW	0x0	UAI: USB Host Asynchronous Interrupt (USBHSTASYNCINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
0	RO	X	NAKI: NAK Interrupt Bit Read Only. This bit is read only. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared. 0 = DISABLE 1 = ENABLE

23.9.1.13 USB2_CONTROLLER_USB2D_USBEXTINTR_0

USB2D EXTINTR Register

Offset: 0x12c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000

Bit	Reset	Description
4	0x0	TIE1: General Purpose Timer Interrupt Enable 1 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
3	0x0	TIE0: General Purpose Timer Interrupt Enable 0 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
2	0x0	UPIE: UPIE Interrupt Enable. 1 = USB controller issues an interrupt if the UPA bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
1	0x0	UAIE: UAIE Interrupt Enable. 1 = USB controller issues an interrupt if the UAI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
0	0x0	NAKE: NAK Interrupt Enable. 1 = USB controller issues an interrupt if the NAKI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE

23.9.1.14 USB2_CONTROLLER_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 0x130 | Read/Write: R/W | Reset: 0bxxxx00000000010000000x1x11x0000000

Bit	R/W	Reset	Description
27:24	RW	0x0	HIRD: Host Initiated Resume Duration RW. Default = 0000b. This has the same behavior as bits 7:4 of the BA field of the HOSTPCx register. When writing to this field all BA[7:4] fields of all HOSTPCx registers will be set to this value. This field is used by system software to specify the minimum amount of time the host controller will drive the K-state during a host-initiated resume from an LPM state (e.g., L1), and is conveyed to each LPM-enabled device (via the HIRD bits within an LPM Tokens bmAttributes field) upon entry into a low-power state. Note the host controller is required to drive resume signaling for at least the amount of time specified in the HIRD value conveyed to the device during any proceeding host-initiated resume. Also note that the host controller is not required to observe this requirement during device-initiated resumes. Encoding for this field is identical to the definition for the similarly named HIRD field within an LPM Token, specifically: a value 0000b equals 50 µs and each additional increment adds 75 µs.

Bit	R/W	Reset	Description
			For example, the value 0001b equals 125 µs, and the value 1111b equals 1,175 µs (~1.2 ms).
23:16	RW	0x8	ITC: Interrupt Threshold Control .Read/Write. Default 08h. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval. 00h: Immediate (no threshold). 01h: 1 micro-frame. 02h: 2 micro-frames. 04h: 4 micro-frames. 08h: 8 micro-frames. 10h: 16 micro-frames. 20h: 32 micro-frames. 40h: 64 micro-frames 0 = IMMEDIATE 2 = ONE_MF 4 = TWO_MF 8 = EIGHT_MF 16 = SIXTEEN_MF 32 = THIRTY_TWO_MF 64 = SIXTY_FOUR_MF
15	RW	0x0	FS2: Bit 2 of Frame List Size.
14	RW	0x0	ATDTW: Add DTD Tripwire. This bit is used as a semaphore when a dTD is added to an active (primed) endpoint. This bit is set and cleared by software and will be cleared by hardware when a hazard exists such that adding a dTD to a primed endpoint may go unnoticed. 0 = CLEAR 1 = SET
13	RW	0x0	SUTW: Setup Tripwire. This bit is used as a semaphore when the 8 bytes of setup data read extracted by the firmware. If the setup lockout mode is off, then there exists a hazard when new setup data arrives and firmware is copying setup data from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists. 0 = CLEAR 1 = SET
11	RW	0x1	ASPE: Asynchronous Schedule Park mode Enable. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. This field is set to "1" in this implementation. 0 = DISABLE 1 = ENABLE
9:8	RW	0x3	ASP1_ASP0: Asynchronous Schedule Park Mode Count (OPTIONAL) Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is RO. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior. This field is set to 3h in this implementation and is Read/Write capable.
7	RO	X	LR: Light Host/Device Controller Reset (OPTIONAL). Read Only. Not Implemented. This field will always be "0".
6	RW	0x0	IAA: Interrupt on Async Advance Doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results. 0 = CLEAR 1 = SET
5	RW	0x0	ASE: Asynchronous Schedule Enable. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
4	RW	0x0	PSE: Periodic Schedule Enable. This bit controls whether the host controller skips processing the Periodic Schedule. 0 = Do not process the Periodic Schedule 1 = Use the PERIODICLISTBASE register to access the Periodic Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
3:2	RW	0x0	FS1_FS0: Frame List Size. (Read/Write). 000 = (Default). This field is Read/Write only if the Programmable Frame List Flag in the HCCPARAMS registers is set to one. Hence this field is Read/Write for this implementation. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3, and 2. 000 = 1024 elements (4096 bytes) Default value 001 = 512 elements (2048 bytes) 010 = 256 elements (1024 bytes) 011 = 128 elements (512 bytes) 100 = 64 elements (256 bytes) 101 = 32 elements (128 bytes) 110 = 16 elements (64 bytes) 111 = 8 elements (32 bytes) Only the host controller uses this field.
1	RW	0x0	RST: Controller Reset. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host Controller: When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller results in undefined behavior. Device Controller: When software writes a one to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. Writing a one to this bit in device mode is not recommended. 0 = CLEAR 1 = SET
0	RW	0x0	RS: Run/Stop: Host Controller: When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HCHalted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the host controller is in the Halted state (i.e., HCHalted in the USBSTS register is a one). Device Controller: Writing a one to this bit will cause the device controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this will cause a detach event. 0 = STOP 1 = RUN

23.9.1.15 USB2_CONTROLLER_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 0x134 | Read/Write: R/W | Reset: 0b00000000000000000x100x0000x0000

Bit	R/W	Reset	Description
31:16	RW	0x0	PPCI: Port-n Change Detect - RW. Default = 0000h. The definition for each bit is identical to the Port Change Detect field (bit 2 of this register) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set to a one then a port change event was detected on Port 2. The N_PORTS field in HCSPARAMS specifies how many ports are exposed by the host controller and thus how many bits in this field are valid.

Bit	R/W	Reset	Description
15	RW	0x0	AS: Asynchronous Schedule Status. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. If AS = ASE: 1= Enable Asynchronous Schedule. 0= Disable Asynchronous Schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
14	RW	0x0	PS: Periodic Schedule Status. This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. If PS = PSE then: 1 = Periodic Schedule is enabled or 0 = Periodic Schedule is disabled. Only used by the host controller. 0 = DISABLE 1 = ENABLE
13	RO	X	RCL: Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	HCH: HCHalted. 1 = Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g., internal error). Only used by the host controller. 0 = UNHALTED 1 = HALTED
11	RW	0x0	UALT_INT: ULPI alt_int Interrupt. 0 = Default. This interrupt bit is set when an RXCMD is received through the ULPI interface with bit 7 set (alt_int).The alt_int itself is set when an unmasked event occurs on any bit in the Carkit Interrupt Latch Register, in the ULPI PHY The software should read the Carkit Interrupt Latch Register (Read to Clear) through the ULPI Viewport to check the source of the interrupt. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1. 0 = NOT_ULPI_ALT_INT 1 = ULPI_ALT_INT
10	RW	0x0	ULPI_INT: ULPI Interrupt. This bit is set whenever an interrupt is received from ULPI PHY. Software writes 1 to clear it. 0 = NOT_ULPI_INT 1 = ULPI_INT
8	RW	0x0	SLI: DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller. 0 = NOTSUSPEND 1 = SUSPENDED
7	RW	0x0	SRI: SOF Received. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When an SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1 ms in device FS mode and every 125 µs in HS mode and will be synchronized to the actual SOF that is received. Since the device controller is initialized to FS before connect, this bit will be set at an interval of 1 ms during the prelude to the connect and chirp. In host mode, this bit will be set every 125 µs and can be used by the host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit. 0 = SOF_NOT_RCVD 1 = SOF_RCVD
6	RW	0x0	URI: USB Reset Received. When the device controller detects a USB Reset and enters the default state, this bit is set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. 0 = NO_USB_RESET 1 = USB_RESET
5	RW	0x0	AAI: Interrupt and Asynchronous Advance. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit

Bit	R/W	Reset	Description
			indicates the assertion of that interrupt source. Only used by the host controller 0 = NOT_ADVANCED 1 = ADVANCED
4	RO	X	SEI: System Error. This bit is not used in this implementation and will always be set to "0". 0 = NO_ERROR 1 = ERROR
3	RW	0x0	FRI: Frame List Rollover. The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [1 3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FHINDEX [12] toggles. Only used by the host controller. 0 = NO_ROLLOVER 1 = ROLLOVER
2	RW	0x0	PCI: Port Change Detect. The Host Controller sets this bit to a 1 when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operational states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits, respectively. This bit is not EHCI compatible. 0 = NO_PORT_CHANGE 1 = PORT_CHANGE
1	RW	0x0	UEI: USB Error Interrupt. This bit gets set by the Host/Device controller when completion of a USB transaction results in an error condition. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. 0 = NO_ERROR 1 = ERROR
0	RW	0x0	UI: USB Interrupt. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) as an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes. 0 = NO_INT 1 = INT

23.9.1.16 USB2_CONTROLLER_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 0x138 | Read/Write: R/W | Reset: 0b0000000000000000xxxx00x000000000

Bit	Reset	Description
31:16	0x0	PPCE: Port-n Change Detect Enable - RW. Default = 0000h. The definition for each bit in this field is identical to bit 2 of this register (Port Change Interrupt Enable) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set (1b) then a port change event was detected on Port 2. When a bit in this field is a one, and the corresponding Port-n Change Detect bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port-n Change Detect bit.
11	0x0	UALTIE: ULPI alt_int Interrupt Enable. 1 = USB controller issues an interrupt if the ULPI_ALT_INT bit in the USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_ALT_INT bit. 0 = DISABLE 1 = ENABLE
10	0x0	ULPIE: ULPI Interrupt Enable. 1 = USB controller issues an interrupt if ULPI_INT bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_INT bit. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	SLE: Sleep Enable. 1 = Device controller issues an interrupt if DCSuspend bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the DCSuspend bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
7	0x0	SRE: SOF Received Enable. 1 = Device controller issues an interrupt if SOF Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the SOF Received bit. 0 = DISABLE 1 = ENABLE
6	0x0	URE: USB Reset Enable.1 = Device controller issues an interrupt if USB Reset Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
5	0x0	AAE: Interrupt on Asynchronous Advance Enable. 1 = The host controller issues an interrupt at the next interrupt threshold if Interrupt on Async Advance bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
4	0x0	SEE: System Error Enable. 1 = Host/device controller issues an interrupt if the System Error bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the System Error bit. 0 = DISABLE 1 = ENABLE
3	0x0	FRE: Frame List Rollover Enable. 1 = Host controller issues an interrupt if Frame List Rollover bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
2	0x0	PCE: Port Change Detect Enable. 1 = Host/device controller issues an interrupt if Port Change Detect bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Port Change Detect bit. 0 = DISABLE 1 = ENABLE
1	0x0	UEE: USB Error Interrupt Enable. 1 = Host controller issues an interrupt at the next interrupt threshold if the USBERRINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register. 0 = DISABLE 1 = ENABLE
0	0x0	UE: USB Interrupt Enable. 1 = Host/device issues an interrupt at the next interrupt threshold if the USBINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBINT bit. 0 = DISABLE 1 = ENABLE

23.9.1.17 USB2_CONTROLLER_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 0x13c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
13:0	X	FRINDEX: Frame Index. The value in this register increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. Each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N 000b (1024) 12 001b (512) 11 010b (256) 10

		011b	(128)	9
		100b	(64)	8
		101b	(32)	7
		110b	(16)	6
		111b	(8)	5
		In device mode, the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current micro-frame.		

23.9.1.18 USB2_CONTROLLER_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 0x144 | Read/Write: R/W | Reset: 0b00000000000000000000xxxxxxxxxxxx

Bit	Reset	Description
31:25	0x0	USBADR: Device mode. The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS request.
24	0x0	USBADRA: Device Address Advance. Default=0. When this bit is 0, any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD cannot write of the device address within 2 ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2 ms USB requirement.
31:12	0x0	BASEADR: Host mode: This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. The HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.

23.9.1.19 USB2_CONTROLLER_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 0x148 | Read/Write: R/W | Reset: 0b00000000000000000000000000xxxxx

Bit	Reset	Description
31:11	0x0	EPBASE: Device mode. This register contains the address of the top of the endpoint list in system memory. These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). Only used by the device controller.
31:5	0x0	ASYBASE: Host mode. This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.

23.9.1.20 USB2_CONTROLLER_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 0x14c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0x

Bit	R/W	Reset	Description
1	RW	0x0	TTAC: Embedded TT Async Buffers Clear. (Read/Write to set). This field will clear all pending transactions in the embedded TT Async Buffer(s). The clear will take as much time as necessary to clear buffer without interfering with a transaction in progress. TTAC will return to zero after being set by software only after the actual clear occurs.
0	RO	X	TTAS: Embedded TT Async Buffers Status. (Read Only). This read only bit will be 1 if one or more transactions are being held in the embedded TT Async. Buffers. When this bit is a zero, then all outstanding transactions in the embedded TT have been flushed.

23.9.1.21 USB2_CONTROLLER_USB2D_BURSTSIZE_0

USB2D Burst Size Register

Offset: 0x150 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000100000001000

Bit	Reset	Description
15:8	0x8	TXPBURST: Programmable TX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.
7:0	0x8	RXPBURST: Programmable RX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory.

23.9.1.22 USB2_CONTROLLER_USB2D_TXFILLTUNING_0

USB2D Transmit Fill Tuning Register

Offset: 0x154 | Read/Write: R/W | Reset: 0bxxxxxxxxxx000010xxx0000000000000

Bit	Reset	Description
21:16	0x2	TXFIFOTHRES: FIFO Burst Threshold. (Read/Write). This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
12:8	0x0	TXSCHHEALTH: Scheduler Health Counter. (Read/Write To Clear) [Default = 0] This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max out at 31.
7:0	0x0	TXSCHOH: Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267 µs when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333 µs when a device is connected in Low/Full Speed Mode

23.9.1.23 USB2_CONTROLLER_USB2D_ICUSB_CTRL_0

This register enables and controls the ICUSB FS/LS transceiver.

USB2D ICUSB control register

Offset: 0x15c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000

Bit	Reset	Description
3	0x0	IC_ENB1: ICUSB transceiver enable. This bit enables the ICUSB transceiver. To enable the interface, the bits PTS must be set to 11 in the PORTSCx. Writing a '1' to this bit selects the IC_USB interface. 0 = DISABLE 1 = ENABLE
2:0	0x0	IC_VDD1: ICUSB voltage select. It selects which voltage is being supplied to the ICUSB peripheral. 000 = No voltage 001 = 1.0V - reserved 010 = 1.2V - reserved 011 = 1.5V - reserved 100 = 1.8V 101 = 3.0V 110 = reserved 111 = reserved The Voltage negotiation should happen between enabling port power (PP) and asserting the run/stop bit in register.

23.9.1.24 USB2_CONTROLLER_USB2D_ULPI_VIEWPORT_0

This register provides indirect access to the ULPI PHY register set. Although the USB controller performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.

Note: WRITES TO THE ULPI THROUGH THE VIEWPORT CAN SUBSTANTIALLY HARM STANDARD USB OPERATIONS. CURRENTLY NO USAGE MODEL HAS BEEN DEFINED WHERE SOFTWARE SHOULD NEED TO EXECUTE WRITES DIRECTLY TO THE ULPI PHY. SEE EXCEPTION REGARDING OPTIONAL FEATURES BELOW.

Note: Executing read operations through the ULPI Viewport should have no harmful side effects to standard USB operations.

There are two operations that can be performed with the ULPI Viewport: wakeup and read /write operations.

The wakeup operation is used to put the ULPI interface into normal operation mode and reenable the clock if necessary. A wakeup operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or carkit mode.

The ULPI state can be determined by reading the sync state bit (ULPI_SYNC_STATE). If this bit is a one, then ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPI_SYNC_STATE indicates a 0 then then read/write operations will not be able to execute. Undefined behavior will result if ULPI_SYNC_STATE = 0 and a read or write operation is performed.

To execute a wakeup operation, write all 32-bits of the ULPI Viewport where ULPI_PORT is constructed appropriately and the ULPI_WAKEUP bit is a 1 and ULPI_RUN bit is a 0. Poll the ULPI Viewport until ULPI_WAKEUP is zero for the operation to complete.

To execute a read or write operation, write all 32 bits of the ULPI Viewport where ULPI_DATA_WR, ULPI_REG_ADDR, ULPI_PORT, ULPI_RD_WR are constructed appropriately and the ULPI_RUN bit is a 1. Poll the ULPI Viewport until ULPI_RUN is zero for the operation to complete. Once ULPI_RUN is zero, the ULPI_DATA_RD will be valid if the operation was a read.

The polling method above can be changed to interrupt driven using the ULPI interrupt defined in the USBSTS and USBINTR registers. When a wakeup or read/write operation is complete, the ULPI_INT interrupt will be set.

There are several optional features that may need to be enabled or disabled by system software as part of system configuration. These bits are contained in the Interface and OTG Control registers of the ULPI PHY register set. These registers also contain bits which are controlled by the link dynamically and therefore should be only modified by system software using the Set/Clear access method. Direct writes to these registers could have harmful side effects to the standard USB operations. The optional bits are as follows: Bits 3 through 7 in the Interface Control register and Bits 6 and 7 in the OTG Control register.

Please refer to the ULPI Specification Revision 1.1 for further information on the use of the optional features.

USB2D ULPI Viewport Register

Offset: 0x160 | Read/Write: R/W | Reset: 0b000xx00000000000xxxxxxxx00000000

Bit	R/W	Reset	Description
31	RW	0x0	ULPI_WAKEUP: ULPI Wakeup. Writing the 1 to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
30	RW	0x0	ULPI_RUN: ULPI read/write Run. Writing the 1 to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
29	RW	0x0	ULPI_RD_WR: ULPI read/write control. (0) Read; (1) Write. This bit selects between running a read or write operation. 0 = READ 1 = WRITE
27	RO	X	ULPI_SYNC_STATE: ULPI sync state. (1) Normal Sync. State. (0) In another state (i.e., carkit, serial, low power). This bit represents the state of the ULPI interface. 0 = NOT_NORMAL 1 = NORMAL
26:24	RW	0x0	ULPI_PORT: ULPI PHY port number. This field should be always written as 0.
23:16	RW	0x0	ULPI_REG_ADDR: ULPI PHY register address. When doing a read or write operation to the ULPI PHY, the address of the ULPI PHY register being accessed is written to this field.
15:8	RO	X	ULPI_DATA_RD: ULPI PHY data read. The data from the ULPI PHY register can be read from here after the read operation completes.
7:0	RW	0x0	ULPI_DATA_WR: ULPI PHY data write. The data to write to the ULPI PHY register is written here.

23.9.1.25 USB2_CONTROLLER_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 0x174 | Read/Write: R/W | Reset: 0b0000000xx0000000xxx1xx00x0xx010x

Bit	R/W	Reset	Description
31:25	RW	0x0	DA: Device Address - RW. Default = 0000000b. The 7-bit USB device address for the device attached to an immediately downstream of the associated root port. A value of zero indicates no device is present or software support for this feature is not present. This is used by the Controller when sending the LPM token. This field is only valid when the core is operating in host mode. If in device mode it will be read only and always equal to 0000000b.

Bit	R/W	Reset	Description
24:23	RO	X	SSTS: Suspend Status - RO. Default = 00b. These two bits are used by software to determine whether an L1-based suspend request was successful, specifically: 00b - L1 state entered with success. ACK received from peripheral. 01b - NYET received from peripheral. It was not able to enter L1 state this time. 10b - L1 state not supported by peripheral. STALL received. 11b - Peripheral did not respond or an error occurred. The value of this field is only valid when the port resides in the L0 state - that is, the meaning of these bits is invalid whenever bit 7 of this register (SUSP) is one. Ideally, the Controller driver should read this register if it receives an interrupt after issuing a suspend using L1 support. In case of a non-success a port change interrupt will be fired and this field should be checked for a possible L1 failure. This field is only valid when the core is operating in host mode. If in device mode, it will be always equal to 00b. 0 = L1STATE_ENTERED 1 = NYET_PERIPH 2 = L1STATE_NOT_SUPPORTED 3 = PERIPH_NORESP_ERR
22	RW	0x0	WKOC: Default = 0b. Wake on Over-current Enable: Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power (PP) is zero. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. 0 = DISABLE 1 = ENABLE
21	RW	0x0	WKDS: Wake on Disconnect Enable: Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power (PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. This bit should not be written to 1 if there is no device connected. After the device disconnect is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
20	RW	0x0	WKCN: Wake on Connect Enable: Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power (PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. This bit should not be written to 1 while the device is connected. After the device connection is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any value other than zero indicates that the port is operating in test mode. Value Specific Test. 0000b: Not enabled. 0001b: J_ STATE. 0010b: K_STATE. 0011b: SEQ_NAK. 0100b: Packet. 0101b: FORCE_ENABLE. 0110b to 1111b: Reserved. Refer to Chapter 7 of the USB Specification Revision 2.0 for details on each test mode. 0 = NORMAL_OP 1 = TEST_J 2 = TEST_K 3 = TEST_SE0_NAK 4 = TEST_PKT 5 = TEST_FORCE_ENABLE
15:14	RO	X	PIC: Port Indicator Control: This field is not supported in the current implementation. Please use a GPIO if you wish to use Port Indicators.
13	RO	X	PO: Port Owner. Port owner handoff is not implemented in this design; therefore this bit will always be 0.
12	RW	0x1	PP: Port Power: The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0b 0b Read Only. A device controller with no OTG capability does not have port power control switches. 1b 1b/0b RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e., PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port).

Bit	R/W	Reset	Description
			0 = NOT_POWERED 1 = POWERED
11:10	RO	X	LS: Line state. These bits reflect the current logical levels of the D+ (bit 10) and D- (bit 11) signal lines. The encoding of the bits is: 00b = SE0. 01b = J-state. 10b = K-state. 11b = Undefined. The value of this field is undefined if Port Power (PP) is zero in host mode. In host mode, the use of line-state by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of line-state by the device controller driver is not necessary. 0 = SE0 1 = J_STATE 2 = K_STATE 3 = UNDEFINED
9	RW	0x0	SLP: Suspend using L1 - RW. Default = 0b. When this bit is set to '1' and a non-zero Device Address (DA) is specified, the Controller will instigate L1 entry during suspend (bit 7) and L1 exit during resume (bit 6). When this bit is set to zero, the Controller will use the legacy (L2) mechanism. Software should only set this bit when the device attached immediately downstream supports L1 transitions. When acting as device, this bit is read-only and set to '1' by the hardware when the Controller enters is L1 state (LPM token received and accepted). Note: HSP is redundant with PSPD[27:26]. This bit is not defined in the EHCI specification.
8	RW	0x0	PR: This field is zero if Port Power (PP) is zero. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. 0 = NOT_USB_RESET 1 = USB_RESET
7	RO	X	SUSP: Port suspend. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend. When in the suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (i.e., Port enabled bit is a zero), the results are undefined. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: Read Only. This bit is a read only status bit. 0 = NOT_SUSPEND 1 = SUSPEND

Bit	R/W	Reset	Description
6	RW	0x0	FPR: Force Port Resume. 1= Resume detected/driven on port. 0=No resume (K state) detected/driven on port. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit remains a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time the resume operation to clear the bit when the port control state switches to HS or FS idle. This field is zero if Port Power (PP) is zero in host mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. Software should ensure that the PHY clock is operational before writing a 1 to this bit to start the resume sequence. This is true for both Device and Host modes. 0 = NO_RESUME 1 = RESUME
5	RO	X	OCC: Over-current Change: Not supported 0 = NO_CHANGE 1 = CHANGE
4	RO	X	OCA: Over-current Active: Not supported 0 = NO_OVER_CURRENT 1 = OVER_CURRENT
3	RW	0x0	PEC: Port Enable/Disable Change: 1=Port enabled/disabled status has changed. 0=No change. In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if Port Power (PP) is zero. In Device mode: The device port is always enabled. (This bit will be zero.) 0 = NO_CHANGE 1 = CHANGE
2	RW	0x1	PE: Port Enabled/Disabled: 1=Enable. 0=Disable (default). In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: The device port is always enabled. (This bit will be one) 0 = PORT_DISABLED 1 = PORT_ENABLED
1	RW	0x0	CSC: Connect Status Change: 1 =Change in Current Connect Status. 0=No change (default) In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power (PP) is zero in host mode. This bit is undefined in device controller mode 0 = NO_CHANGE 1 = CHANGE

Bit	R/W	Reset	Description
0	RO	X	CCS: Current Connect Status. In Host Mode: 1=Device is present on port. 0=No device is present (default). This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: 1=Attached 0=Not Attached (default) A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0 = NOT_CONNECTED 1 = CONNECTED

23.9.1.26 USB2_CONTROLLER_USB2D_HOSTPC1_DEVLC_0

USB2D Device Mode LPM Behavior and Control Register

This register controls the LPM behavior and controls the behavior of each USB port. Only available in device mode. This register shares the same address with the HOSTPC1 register (which is used only in host mode).

USB2D Host Mode LPM Behavior and Control Register

This register controls the LPM behavior and controls the behavior of each USB port. Only available in host mode. This register shares the same address with the DEVLC register (which is used only in device mode).

Offset: 0x1b4 | Read/Write: R/W | Reset: 0b0000xxx0000000000000xxxxxxxxxxx0

Bit	R/W	Reset	Description
31:29	RW	0x0	PTS: Parallel transceiver select. This bit is not defined in the EHCI specification. 0 = UTMI 1 = RESERVED 2 = ULPI 3 = ICUSB_SER 4 = HSIC
28	RW	0x0	STS: Serial transceiver not selected. This is the only value supported. This bit is not defined in the EHCI specification. 0 = PARALLEL_IF 1 = SERIAL_IF
27	RO	X	PTW: Parallel Transceiver Width. Fixed to 0. This bit is not defined in the EHCI specification. 0 = EIGHT_BIT 1 = RESERVED
26:25	RO	X	PSPD: This register field indicates the speed at which the port is operating. 00 = Full Speed 01 = Low Speed 10 = High Speed. This bit is not defined in the EHCI specification. 0 = FULL_SPEED 1 = LOW_SPEED 2 = HIGH_SPEED 3 = RESERVED
24	RW	0x0	ALPD: Auto Low Power While Disconnect - RW. Default = 0b. If set, this feature will be enabled, and every time the port enters the disconnect state it will also enter low power state,disabling the transceiver clock. The behavior will be same as if the PHCD bit was enabled (in fact this bit will be set to 1 as soon as low power mode is enabled).When this field is set the WKCN field of PORTSCx register (Wake on Connect Enable) will also be set. This way the core will wake up in case a connect is detected. There will be a delay between the detection of a disconnect and actually enter in low power mode. This delay can be controlled by writing to the ALPDD field in the register USBMODE. 0 = DONT_AUTO_LOW_POWER_WHILE_DISCONNECT 1 = AUTO_LOW_POWER_WHILE_DISCONNECT

Bit	R/W	Reset	Description
23	RW	0x0	PFSC: Port Force Full Speed Connect - RW. Default = 0b. Writing this bit to a '1b' will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with an HS host, hub or device. This bit is not defined in the EHCI specification. 0 = DONT_FORCE_FULL_SPEED 1 = FORCE_FULL_SPEED
22	RW	0x0	PHCD: PHY Low Power Suspend - Clock disable: Writing this bit to a 1 will disable the PHY clock. Write a 0 enables it. Reading this bit will indicate the status of the PHY clock. NOTE: The PHY clock cannot be disabled if it is being used as the system clock. In device mode, the PHY can be put into Low Power Clock Disable when the device is not running (USBCMD RS=0b) or the host has signaled suspend (PORTSCx SUSP=1b). Low Power Clock Disable will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the Controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low Power Clock Disable is completely under the control of software. 0 = DISABLE 1 = ENABLE
21:20	RW	0x0	LPMX: Auto LPM set - RW. Default = 00b. This bit field is valid during Host Mode Only. For Device Mode, this bit is reserved. Value Meaning 00b Disables auto LPM. 01b If there is no activity for a certain number of SOFs the controller will send an LPM token, enter in suspend and issue a port change interrupt. 10b Same as above but without issuing an interrupt. 11b Reserved for futures LPM enhancements. The detection of no activity is based on the absence of response from the downstream device. Because of this limitation this feature should not be used if ISO OUT endpoints are being used (as no handshake is expected).
17	RW	0x0	ASUS: Auto Low Power - RW. Default = 0b. This bit field is valid during Device Mode Only. In Host Mode, it is part of the ELPM field. This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters in suspend state it will also enter in low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	EPLPM: Endpoint for LPM token - RW. Default = 0000b. This bit field is valid during Host Mode Only. For Device Mode, bits [19:18] are reserved and bit 17 is ASUS, while bit 16 is STL. This sets the endpoint number to which the LPM token will be sent. It will be directly mapped to the ENDP field in the LPM token with the EXT PID.
16	RW	0x0	STL: STALL reply to LPM token - RW. Default = 0b. This bit field is valid during Device Mode Only. In Host Mode, it is part of the ELPM field. When this bit is set to '1', the Controller will reply always with STALL to all incoming LPM tokens. This bit overrides the LPM NYET bit (NYT). 0 = DISABLE 1 = ENABLE
15:12	RW	0x0	LPMFRM: Auto LPM SOF Threshold - RW. Default = 0000b. This bit field is valid during Host Mode Only. For Device Mode, this field is reserved. This holds the SOF counter threshold. When the number of SOFs with no activity in between reaches this threshold and the auto LPM is enabled, an LPM token will be sent and the port will enter in suspend state. The SOF counter for this threshold is incremented each 125 µs, even if the port is not in HS operation.
11:1	RO	X	BA: bmAttributes - RO. Default = 00000000000b. This holds the bmAttributes field of the LPM sub-token received, after the EXT PID token.

Bit	R/W	Reset	Description
0	RW	0x0	NYT_ASUS: NYET reply to LPM token - RW. Default = 0b. This bit is NYT during Device Mode. Details follow: When this bit is to '1', the device controller will NYET all the LPM tokens. When this bit is set to '0', the Controller will ACK all the LPM tokens if the STALL bit (STL) is also set to '0'. This bit is ASUS in Host Mode. Details follow: This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters in suspend state it will also enter in low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE

23.9.1.27 USB2_CONTROLLER_USB2D_OTGSC_0

USB2D On-The-Go (OTG) Status and Control Register

Offset: 0x1f4 | Read/Write: R/W | Reset: 0bx0000000x0000000xxxxxxxxxx000x00

Bit	R/W	Reset	Description
30	RW	0x0	DPIE: Data Pulse Interrupt Enable. Setting this bit enables the Data pulse interrupt. 0 = DISABLE 1 = ENABLE
29	RW	0x0	ONEMSE: 1 millisecond timer Interrupt enable. Setting this bit enables the 1 millisecond timer interrupt. 0 = DISABLE 1 = ENABLE
28	RW	0x0	BSEIE: B Session End Interrupt Enable. Setting this bit enables the B session end interrupt 0 = DISABLE 1 = ENABLE
27	RW	0x0	BSVIE: B Session Valid Interrupt Enable. Setting this bit enables the B session valid interrupt 0 = DISABLE 1 = ENABLE
26	RW	0x0	ASVIE: A Session Valid Interrupt Enable. Setting this bit enables the A session valid interrupt 0 = DISABLE 1 = ENABLE
25	RW	0x0	AVVIE: A VBus Valid Interrupt Enable. Setting this bit enables the A VBus valid interrupt 0 = DISABLE 1 = ENABLE
24	RW	0x0	IDIE: USB ID Interrupt Enable. Setting this bit enables the USB ID interrupt 0 = DISABLE 1 = ENABLE
22	RW	0x0	DPIS: Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0). PortPower = Off (0). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
21	RW	0x0	ONEMSS: 1 millisecond timer Interrupt Status: This bit is set once every millisecond. Software writes a 1 to clear it. 0 = INT_CLEAR 1 = INT_SET
20	RW	0x0	BSEIS: B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software writes a 1 to clear this bit . 0 = INT_CLEAR 1 = INT_SET
19	RW	0x0	BSVIS: B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET

Bit	R/W	Reset	Description
18	RW	0x0	ASVIS: A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software writes a one to clear this bit. 0 = INT_CLEAR 1 = INT_SET
17	RW	0x0	AVVIS: A VBus Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
16	RW	0x0	IDIS: USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
14	RO	X	DPS: Data Bus Pulsing Status. A 1 indicates data bus pulsing is being detected on the port. 0 = STS_CLEAR 1 = STS_SET
13	RO	X	ONEMST: 1 millisecond timer toggle. This bit toggles once per millisecond 0 = STS_CLEAR 1 = STS_SET
12	RO	X	BSE: B session End. Indicates VBus is below the B session end threshold 0 = STS_CLEAR 1 = STS_SET
11	RO	X	BSV: B Session Valid. Indicates VBus is above the B session valid threshold 0 = STS_CLEAR 1 = STS_SET
10	RO	X	ASV: A Session Valid. Indicates VBus is above the A session valid threshold 0 = STS_CLEAR 1 = STS_SET
9	RO	X	AVV: A VBus Valid. Indicates VBus is above the A VBus valid threshold 0 = STS_CLEAR 1 = STS_SET
8	RO	X	ID: USB ID: 0 = A-device 1 = B-device 0 = A_DEV 1 = B_DEV
5	RW	0x0	IDPU: USB ID Pullup 0 = CLEAR 1 = SET
4	RW	0x0	DP: Data Pulsing. Setting this bit causes the pull-up on DP to be asserted for data pulsing during SRP. 0 = NO_DATA_PULSE 1 = DATA_PULSE
3	RW	0x0	OT: OTG Termination. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM. 0 = NO_OTG_TERM 1 = OTG_TERM
1	RW	0x0	VC: VBUS Charge. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP. 0 = NO_VBUS_CHRG 1 = VBUS_CHRG
0	RW	0x0	VD: VBUS_Discharge. Read/write. Setting this bit causes Vbus to discharge through a resistor. 0 = NO_VBUS_DISCHRG 1 = VBUS_DISCHRG

23.9.1.28 USB2_CONTROLLER_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 0x1f8 | Read/Write: R/W | Reset: 0b00000000000000000xxxxxxxxx0xxxxx

Bit	R/W	Reset	Description
31:16	RW	0x0	ALPDD: Auto Low Power While Disconnect Delay, Used when the ALPD field of the HOSTPCx register is set. Defines the delay between disconnect detection and entering in low power mode. The delay is <this register value>*64*100/3 in milliseconds. The maximum value is 65535*64*100/3 = 139,808 ms. The minimum value is 0. Only used in host mode.
15	RW	0x0	SRT: Shorten USB Reset Time. Software should never set this to 1.
5	RW	0x0	VBPS: VBUS Power Select This can be used by logic that selects between an on-chip Vbus power source (charge pump) and an off-chip source in systems when both are available. Only to be used in Host Mode. No functionality is implemented for this, so software should not use this bit.
4	RO	X	SDIS: Stream disable: 1: Streaming is disabled - helpful to avoid overruns/underruns when the system load is too high. 0 = STREAM_ENABLE 1 = STREAM_DISABLE
3	RO	X	SLOM: Setup Lockout Mode: In device mode, this bit controls the behavior of the setup lockout mechanism. 0: Setup lockout is ON (default). 1: Setup lockout is OFF. Firmware requires the use of setup tripwire semaphore in USB2D_USBCMD register. 0 = LOCKOUT_ON 1 = LOCKOUT_OFF
2	RO	X	ES: Endian Select: Note: For this implementation, this bit should be always set to 0 (little endian). 0 = LITTLE_ENDIAN 1 = RESERVED
1:0	RO	X	CM: Controller Mode: The controller mode will default to an idle state and will need to be initialized to the desired operating mode after reset. This register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register. 00 = Idle [Default]. 01 = Reserved. 10 = Device Controller. 11 = Host Controller. 0 = IDLE 1 = RESERVED 2 = DEVICE_MODE 3 = HOST_MODE

23.9.1.29 USB2_CONTROLLER_USB2D_ENDPTNAK_0

USB2D Endpoint NAK register

Offset: 0x200 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTN: TX Endpoint NAK R/WC. Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. 0 = CLEAR 1 = SET
15:0	0x0	EPRN: RX Endpoint NAK R/WC. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. 0 = CLEAR 1 = SET

23.9.1.30 USB2_CONTROLLER_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable Register

Offset: 0x204 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTNE: TX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE
15:0	0x0	EPRNE: RX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE

23.9.1.31 USB2_CONTROLLER_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 0x208 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15	0x0	ENDPTSETUPSTAT15: Endpoint 15 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
14	0x0	ENDPTSETUPSTAT14: Endpoint 14 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
13	0x0	ENDPTSETUPSTAT13: Endpoint 13 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
12	0x0	ENDPTSETUPSTAT12: Endpoint 12 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
11	0x0	ENDPTSETUPSTAT11: Endpoint 11 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
10	0x0	ENDPTSETUPSTAT10: Endpoint 10 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
9	0x0	ENDPTSETUPSTAT9: Endpoint 9 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
8	0x0	ENDPTSETUPSTAT8: Endpoint 8 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
7	0x0	ENDPTSETUPSTAT7: Endpoint 7 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
6	0x0	ENDPTSETUPSTAT6: Endpoint 6 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
5	0x0	ENDPTSETUPSTAT5: Endpoint 5 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
4	0x0	ENDPTSETUPSTAT4: Endpoint 4 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
3	0x0	ENDPTSETUPSTAT3: Endpoint 3 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
2	0x0	ENDPTSETUPSTAT2: Endpoint 2 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
1	0x0	ENDPTSETUPSTAT1: Endpoint 1 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
0	0x0	ENDPTSETUPSTAT0: Endpoint 0 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

23.9.1.32 USB2_CONTROLLER_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 0x20c | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	PETB15: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
30	0x0	PETB14: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
29	0x0	PETB13: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
28	0x0	PETB12: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
27	0x0	PETB11: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
26	0x0	PETB10: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
25	0x0	PETB9: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
24	0x0	PETB8: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
23	0x0	PETB7: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
22	0x0	PETB6: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
21	0x0	PETB5: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
20	0x0	PETB4: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
19	0x0	PETB3: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
18	0x0	PETB2: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
17	0x0	PETB1: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
16	0x0	PETB0: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
15	0x0	PERB15: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
14	0x0	PERB14: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
13	0x0	PERB13: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
12	0x0	PERB12: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
11	0x0	PERB11: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
10	0x0	PERB10: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
9	0x0	PERB9: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
8	0x0	PERB8: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
7	0x0	PERB7: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
6	0x0	PERB6: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
5	0x0	PERB5: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
4	0x0	PERB4: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
3	0x0	PERB3: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
2	0x0	PERB2: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
1	0x0	PERB1: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
0	0x0	PERB0: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

23.9.1.33 USB2_CONTROLLER_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 0x210 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	FETB15: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
30	0x0	FETB14: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
29	0x0	FETB13: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
28	0x0	FETB12: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
27	0x0	FETB11: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
26	0x0	FETB10: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
25	0x0	FETB9: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
24	0x0	FETB8: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
23	0x0	FETB7: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
22	0x0	FETB6: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
21	0x0	FETB5: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
20	0x0	FETB4: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
19	0x0	FETB3: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
18	0x0	FETB2: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
17	0x0	FETB1: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
16	0x0	FETB0: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
15	0x0	FERB15: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
14	0x0	FERB14: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
13	0x0	FERB13: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
12	0x0	FERB12: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
11	0x0	FERB11: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
10	0x0	FERB10: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
9	0x0	FERB9: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
8	0x0	FERB8: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
7	0x0	FERB7: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
6	0x0	FERB6: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
5	0x0	FERB5: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
4	0x0	FERB4: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
3	0x0	FERB3: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
2	0x0	FERB2: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
1	0x0	FERB1: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
0	0x0	FERB0: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

23.9.1.34 USB2_CONTROLLER_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 0x214 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	ETBR15: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
30	X	ETBR14: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
29	X	ETBR13: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
28	X	ETBR12: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
27	X	ETBR11: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
26	X	ETBR10: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
25	X	ETBR9: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
24	X	ETBR8: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
23	X	ETBR7: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
22	X	ETBR6: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
21	X	ETBR5: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
20	X	ETBR4: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
19	X	ETBR3: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
18	X	ETBR2: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
17	X	ETBR1: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
16	X	ETBR0: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
15	X	ERBR15: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
14	X	ERBR14: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
13	X	ERBR13: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
12	X	ERBR12: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
11	X	ERBR11: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
10	X	ERBR10: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
9	X	ERBR9: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
8	X	ERBR8: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
7	X	ERBR7: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
6	X	ERBR6: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
5	X	ERBR5: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
4	X	ERBR4: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
3	X	ERBR3: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
2	X	ERBR2: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
1	X	ERBR1: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
0	X	ERBR0: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

23.9.1.35 USB2_CONTROLLER_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 0x218 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ETCE15: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
30	0x0	ETCE14: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
29	0x0	ETCE13: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
28	0x0	ETCE12: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
27	0x0	ETCE11: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
26	0x0	ETCE10: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
25	0x0	ETCE9: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
24	0x0	ETCE8: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
23	0x0	ETCE7: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
22	0x0	ETCE6: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
21	0x0	ETCE5: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
20	0x0	ETCE4: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
19	0x0	ETCE3: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
18	0x0	ETCE2: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
17	0x0	ETCE1: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
16	0x0	ETCE0: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
15	0x0	ERCE15: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
14	0x0	ERCE14: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
13	0x0	ERCE13: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
12	0x0	ERCE12: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
11	0x0	ERCE11: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
10	0x0	ERCE10: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
9	0x0	ERCE9: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
8	0x0	ERCE8: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
7	0x0	ERCE7: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
6	0x0	ERCE6: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
5	0x0	ERCE5: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
4	0x0	ERCE4: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
3	0x0	ERCE3: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
2	0x0	ERCE2: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
1	0x0	ERCE1: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
0	0x0	ERCE0: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

23.9.1.36 USB2_CONTROLLER_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 0x21c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23	X	TXE: TX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
19:18	X	TXT: TX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
16	X	TXS: TX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL
7	X	RXE: RX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
3:2	X	RXT: RX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
0	X	RXS: RX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL

23.9.1.37 USB2_CONTROLLER_USB2D_ENDPTCTRL1_0

USB2D Endpoint Control 1 Register

Offset: 0x220 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR

Bit	R/W	Reset	Description
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.38 USB2_CONTROLLER_USB2D_ENDPTCTRL2_0

USB2D Endpoint Control 2 Register

Offset: 0x224 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ

Bit	R/W	Reset	Description
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.39 USB2_CONTROLLER_USB2D_ENDPTCTRL3_0

USB2D Endpoint Control 3 Register

Offset: 0x228 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ

Bit	R/W	Reset	Description
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.40 USB2_CONTROLLER_USB2D_ENDPTCTRL4_0

USB2D Endpoint Control 4 Register

Offset: 0x22c | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.41 USB2_CONTROLLER_USB2D_ENDPTCTRL5_0

USB2D Endpoint Control 5 Register

Offset: 0x230 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.42 USB2_CONTROLLER_USB2D_ENDPTCTRL6_0

USB2D Endpoint Control 6 Register

Offset: 0x234 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.43 USB2_CONTROLLER_USB2D_ENDPTCTRL7_0

USB2D Endpoint Control 7 Register

Offset: 0x238 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.44 USB2_CONTROLLER_USB2D_ENDPTCTRL8_0

USB2D Endpoint Control 8 Register

Offset: 0x23c | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.45 USB2_CONTROLLER_USB2D_ENDPTCTRL9_0

USB2D Endpoint Control 9 Register

Offset: 0x240 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.46 USB2_CONTROLLER_USB2D_ENDPTCTRL10_0

USB2D Endpoint Control 10 Register

Offset: 0x244 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.47 USB2_CONTROLLER_USB2D_ENDPTCTRL11_0

USB2D Endpoint Control 11 Register

Offset: 0x248 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.48 USB2_CONTROLLER_USB2D_ENDPTCTRL12_0

USB2D Endpoint Control 12 Register

Offset: 0x24c | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.49 USB2_CONTROLLER_USB2D_ENDPTCTRL13_0

USB2D Endpoint Control 13 Register

Offset: 0x250 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.50 USB2_CONTROLLER_USB2D_ENDPTCTRL14_0

USB2D Endpoint Control 14 Register

Offset: 0x254 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.1.51 USB2_CONTROLLER_USB2D_ENDPTCTRL15_0

USB2D Endpoint Control 15 Register

Offset: 0x258 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.2 USB1 Controller Interface Registers

These are used to generate the actual RTL registers for USB1 controller interface.

23.9.2.1 USB1_IF_USB_SUSP_CTRL_0

This register controls the suspend and resume behavior of USB controller/PHY.

USB Suspend Control Register

Offset: 0x400 | Read/Write: R/W | Reset: 0bxxxxx00xxxxxx000xxx01000xx000000

Bit	R/W	Reset	Description
26	RW	0x0	FAST_WAKEUP_RESP: Enable Fast Response from UTMIP PHY for a Remote Wakeup request from device. This is used only for cases where wakeup response needs to be within 1 ms of spec. Used for Host mode ONLY. 0 = DISABLE 1 = ENABLE
25	RW	0x0	UTMIP_SUSPL1_SET: Enable SuspendL1 for UTMIP PHY. Enabling this will only cut off clocks to the UTMIP logic. The USB PLLs, PllU and UTMIP PLL will still be running. 0 = DISABLE 1 = ENABLE
18:16	RW	0x0	USB_WAKEUP_DEBOUNCE_COUNT: USB PHY wakeup debounce counter. USB will debounce any wakeup event by the number of clocks programmed in this counter. A value of 0 results in no debounce.
12	RW	0x0	UTMIP_PHY_ENB: Enable UTMIP PHY mode. Set this to 1 if using UTMIP PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
11	RW	0x1	UTMIP_RESET: Reset going to UTMIP PHY (active high). This should be set to 1 whenever programming the UTMIP config registers. It should be cleared to 0 after the programming of UTMIP config registers is done. UTMIP config registers should be programmed only once before doing any transactions on USB. The UTMIP PHY registers should be programmed while UTMIP is in reset. 0 = DISABLE 1 = ENABLE
10	RW	0x0	USB_SUSP_POL: Polarity of the suspend signal going to USB PHY. 0 = Active low (default). 1 = Active high. This should not be changed by software. 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
9	RW	0x0	USB_PHY_CLK_VALID_INT_ENB: USB PHY clock valid interrupt enable. If this bit is enabled, interrupt is generated whenever USB clocks are resumed from a suspend. 0 = DISABLE 1 = ENABLE
8	RO	0x0	USB_PHY_CLK_VALID_INT_STS: USB PHY clock valid interrupt status. This bit is set whenever the USB PHY clock is woken up from suspend. Software must write a 1 to clear this bit. 0 = UNSET 1 = SET
7	RO	X	USB_PHY_CLK_VALID: USB PHY clock valid status. This bit indicates whether the USB PHY is generating a valid clock to the USB controller. If USB PHY clock is running, this bit is set to 1, else it is set to 0. 0 = UNSET 1 = SET
6	RO	X	USB_CLKEN: USB AHB clock enable status. Indicates whether the AHB clock to the USB controller is enabled or not. If AHB clock to USB controller is enabled, this bit is set to 1, else it is set to 0. NOTE: even when this is set to 0, all essential blocks that are required to resume USB clocks from suspend will be active and their AHB clock will not be suspended. 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
5	RW	0x0	USB_SUSP_CLR: Suspend Clear. Software must write a 1 to this bit to bring the PHY out of suspend mode. This is used when the software stops the PHY clock during suspend and then wants to initiate a resume. Software should also write 0 to clear it. NOTE: It is required that software generate a positive pulse on this bit to guarantee proper operation. 0 = UNSET 1 = SET
4	RW	0x0	USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect Enable (device mode). When enabled (1), the USB device will wake up from suspend on a disconnect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
3	RW	0x0	USB_WAKE_ON_CNNT_EN_DEV: Wake on Connect Enable (device mode). When enabled (1), USB device will wake up from suspend on a connect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable. If this bit is enabled, USB will wake up from suspend whenever a resume event is detected on USB. This is valid for both USB device and USB host modes. 0 = DISABLE 1 = ENABLE
1	RW	0x0	USB_WAKEUP_INT_ENB: USB wakeup interrupt enable. If this bit is enabled, interrupt is generated whenever USB wakeup event is generated. 0 = DISABLE 1 = ENABLE
0	RO	0x0	USB_WAKEUP_INT_STS: USB wakeup interrupt status. This bit is set whenever USB wakes up from suspend (a wakeup event is generated). Software must write a 1 to clear this bit. Note that during the wakeup sequence, PHY clocks will be resumed from suspend. Software can check when the PHY clocks are resumed by reading the bit USB_PHY_CLK_VALID. There is also a separate interrupt generated when PHY clock is resumed if USB_PHY_CLK_VALID_INT_EN is set. During the wakeup sequence, first USB_WAKEUP_INT_STS will be set, and it will take some time for the PHY clock to resume, which can be detected by checking USB_PHY_CLK_VALID. 0 = UNSET 1 = SET

23.9.2.2 USB1_IF_USB_PHY_VBUS_SENSORS_0

This register controls the OTG VBUS sensors in the USB PHY. There are 4 VBUS sensors:

- A_VBUS_VLD
- A_SESS_VLD
- B_SESS_VLD
- B_SESS_END

The debounced status of each sensor can be read from the corresponding _STS bit field of the sensor in this register. The _CHG_DET field is set to 1 whenever a change is detected in the value of the _STS bit field of the corresponding sensor. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/COP by appropriately writing the USBD bits in the interrupt controller registers.

In case software wants to override the value for a sensor, it can set the corresponding _SW_EN to 1, and set the corresponding sensor _SW_VALUE to 1 or 0 as per the requirement.

There are two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

USB PHY VBUS SENSORS Control Register

Offset: 0x404 | Read/Write: R/W | Reset: 0bx0000x00x0000x00x0000x00x0000x00

Bit	R/W	Reset	Description
30	RW	0x0	A_VBUS_VLD_WAKEUP_EN: A_VBUS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on A_VBUS_VLD. 0 = DISABLE 1 = ENABLE
29	RW	0x0	A_VBUS_VLD_DEB_SEL_B: A_VBUS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	A_VBUS_VLD_SW_VALUE: A_VBUS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_VBUS_VLD status. This is only valid when A_VBUS_VLD_SW_EN is set. 0 = UNSET 1 = SET
27	RW	0x0	A_VBUS_VLD_SW_EN: A_VBUS_VLD software enable. Enable Software Controlled A_VBUS_VLD. Software sets this bit to drive the value in A_VBUS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
26	RO	X	A_VBUS_VLD_STS: A_VBUS_VLD status. This is set to 1 whenever A_VBUS_VLD sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	A_VBUS_VLD_CHG_DET: A_VBUS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_VBUS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
24	RW	0x0	A_VBUS_VLD_INT_EN: A_VBUS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever A_VBUS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
22	RW	0x0	A_SESS_VLD_WAKEUP_EN: A_SESS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on A_SESS_VLD. 0 = DISABLE 1 = ENABLE
21	RW	0x0	A_SESS_VLD_DEB_SEL_B: A_SESS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
20	RW	0x0	A_SESS_VLD_SW_VALUE: A_SESS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_SESS_VLD status. This is only valid when A_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
19	RW	0x0	A_SESS_VLD_SW_EN: A_SESS_VLD software enable. Enable Software Controlled A_SESS_VLD. Software sets this bit to drive the value in A_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
18	RO	X	A_SESS_VLD_STS: A_SESS_VLD status. This is set to 1 whenever A_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	A_SESS_VLD_CHG_DET: A_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
16	RW	0x0	A_SESS_VLD_INT_EN: A_SESS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever A_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
14	RW	0x0	B_SESS_VLD_WAKEUP_EN: B_SESS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on B_SESS_VLD. 0 = DISABLE 1 = ENABLE
13	RW	0x0	B_SESS_VLD_DEB_SEL_B: B_SESS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	B_SESS_VLD_SW_VALUE: B_SESS_VLD software value. Software should write the appropriate value (1/0) to set/unset the B_SESS_VLD status. This is only valid when B_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	B_SESS_VLD_SW_EN: B_SESS_VLD software enable. Enable Software Controlled B_SESS_VLD. Software sets this bit to drive the value in B_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	B_SESS_VLD_STS: B_SESS_VLD status. This is set to 1 whenever B_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	B_SESS_VLD_CHG_DET: B_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
8	RW	0x0	B_SESS_VLD_INT_EN: B_SESS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever B_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
6	RW	0x0	B_SESS_END_WAKEUP_EN: B_SESS_END wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on B_SESS_END. 0 = DISABLE 1 = ENABLE
5	RW	0x0	B_SESS_END_DEB_SEL_B: B_SESS_END debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B

Bit	R/W	Reset	Description
4	RW	0x0	B_SESS_END_SW_VALUE: B_SESS_END software value. Software should write the appropriate value (1/0) to set/unset the B_SESS_END status. This is only valid when B_SESS_END_SW_EN is set. 0 = UNSET 1 = SET
3	RW	0x0	B_SESS_END_SW_EN: B_SESS_END software enable. Enable Software Controlled B_SESS_END Software sets this bit to drive the value in B_SESS_END_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
2	RO	X	B_SESS_END_STS: B_SESS_END status. This is set to 1 whenever B_SESS_END sensor output is 1. 0 = UNSET 1 = SET
1	RO	0x0	B_SESS_END_CHG_DET: B_SESS_END change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_END. software writes a 1 to clear it 0 = UNSET 1 = SET
0	RW	0x0	B_SESS_END_INT_EN: B_SESS_END interrupt enable. If this field is set to 1, an interrupt is generated whenever B_SESS_END_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE

23.9.2.3 USB1_IF_USB_PHY_VBUS_WAKEUP_ID_0

This register controls the battery charger (VDCD_DET, VDAT_DET), VBUS_WAKEUP and ID sensors. The following sensors are in this register:

- VBUS_WAKEUP
- ID
- VDAT_DET
- VDCD_DET

The debounced status of each sensor can be read from the corresponding _STS bit field of the sensor in this register. The _CHG_DET field is set to 1 whenever a change is detected in the value of the _STS bit field of the corresponding sensor. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/COP by appropriately writing the USBD bits in the interrupt controller registers.

In case software wants to override the value for a sensor, it can set the corresponding _SW_EN to 1, and set the corresponding sensor _SW_VALUE to 1 or 0 as per the requirement.

There are two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

There are debouncers for VDAT_DET and VDCD_DET. These use separate debouncers -CHRG_DEBOUNCE_PERIOD_A and CHRG_DEBOUNCE_PERIOD_B. The debounce values for them are controlled by the register UTMIP_CHRG_DEB_CFG0, fields UTMIP_CHRG_DEBOUNCE_PERIOD_A and UTMIP_CHRG_DEBOUNCE_PERIOD_B. For each sensor, we can select whether to use CHRG_DEBOUNCE_PERIOD_A or CHRG_DEBOUNCE_PERIOD_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

USB PHY VBUS Wakeup and ID Control Register

Offset: 0x408 | Read/Write: R/W | Reset: 0bx0000x00xx000x00xx000x00x1000x00

Bit	R/W	Reset	Description
30	RW	0x0	VBUS_WAKEUP_WAKEUP_EN: VBUS_WAKEUP wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on VBUS_WAKEUP. 0 = DISABLE 1 = ENABLE
29	RW	0x0	VDCD_DET_DEB_SEL_B: VCDT_DET debounce A/B select. Selects the debounce value from UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B from the register UTMIP_CHRG_DEB_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	VDCD_DET_SW_VALUE: VDCD_DET software value. Software should write the appropriate value (1/0) to set/unset the VDCD_DET status. This is only valid when VDCD_DET_SW_EN is set. 0 = UNSET 1 = SET
27	RW	0x0	VDCD_DET_SW_EN: VDCD_DET software enable. Enable Software Controlled VDCD_DET. Software sets this bit to drive the value in VDCD_DET_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
26	RO	X	VDCD_DET_STS: VDCD_DET status. This is set to 1 whenever VDCD_DET sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	VDCD_DET_CHG_DET: VDCD_DET change detect. This field is set by hardware whenever a change is detected in the value of VDCD_DET. software writes a 1 to clear it 0 = UNSET 1 = SET
24	RW	0x0	VDCD_DET_INT_EN: VDCD_DET interrupt enable. If this field is set to 1, an interrupt is generated whenever VDCD_DET_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
21	RW	0x0	VDAT_DET_DEB_SEL_B: VDAT_DET debounce A/B select. Selects between the two debounce values UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
20	RW	0x0	VDAT_DET_SW_VALUE: VDAT_DET software value. Software should write the appropriate value (1/0) to set/unset the VDAT_DET status. This is only valid when VDAT_DET_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	VDAT_DET_SW_EN: VDAT_DET software enable. Enable Software Controlled VDAT_DET. Software sets this bit to drive the value in VDAT_DET_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
18	RO	X	VDAT_DET_STS: VDAT_DET status. This is set to 1 whenever the VDAT_DET sensor output is 1. 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
17	RO	0x0	VDAT_DET_CHG_DET: VDAT_DET change detect. This field is set by hardware whenever a change is detected in the value of VDAT_DET. software writes a 1 to clear it 0 = UNSET 1 = SET
16	RW	0x0	VDAT_DET_INT_EN: VDAT_DET interrupt enable. If this field is set to 1, an interrupt is generated whenever VDAT_DET_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
13	RW	0x0	VBUS_WAKEUP_DEB_SEL_B: VBUS_WAKEUP debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	VBUS_WAKEUP_SW_VALUE: VBUS wakeup software value. Software should write the appropriate value (1/0) to set/unset the VBUS_WAKEUP status. This is only valid when VBUS_WAKEUP_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	VBUS_WAKEUP_SW_EN: VBUS wakeup software enable. Enable Software Controlled VBUS_WAKEUP. Software sets this bit to drive the value in VBUS_WAKEUP_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	VBUS_WAKEUP_STS: VBUS wakeup status. This is set to 1 whenever VBUS_WAKEUP sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	VBUS_WAKEUP_CHG_DET: VBUS wakeup change detect. This field is set by hardware whenever a change is detected in the value of VBUS_WAKEUP. software writes a 1 to clear it 0 = UNSET 1 = SET
8	RW	0x0	VBUS_WAKEUP_INT_EN: VBUS wakeup interrupt enable. If this field is set to 1, an interrupt is generated whenever VBUS_WAKEUP_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
7	RO	X	STATIC_GPI: Static GPI status 0 = UNSET 1 = SET
6	RW	0x1	ID_PU: ID pullup enable. Set to 1. 0 = DISABLE 1 = ENABLE
5	RW	0x0	ID_DEB_SEL_B: ID debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
4	RW	0x0	ID_SW_VALUE: ID software value. Software should write the appropriate value (1/0) to set/unset the ID status. This is only valid when ID_SW_EN is set. 0 = UNSET 1 = SET
3	RW	0x0	ID_SW_EN: ID software enable. Enable Software Controlled ID. Software sets this bit to drive the value in ID_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
2	RO	X	ID_STS: ID status. This is set to 1 whenever ID sensor output is 1. 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
1	RO	0x0	ID_CHG_DET: ID change detect. This field is set by hardware whenever a change is detected in the value of ID. software writes a 1 to clear it 0 = UNSET 1 = SET
0	RW	0x0	ID_INT_EN: ID interrupt enable. If this field is set to 1, an interrupt is generated whenever ID_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE

23.9.2.4 USB1_IF_USB_PHY_ALT_VBUS_STS_0

USB PHY Alternate VBUS/ID Status Register

Offset: 0x40c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
14	X	ID_DIG_C_ALT: IDDIG_C alternate status 0 = UNSET 1 = SET
13	X	ID_DIG_C: IDDIG_C status 0 = UNSET 1 = SET
12	X	ID_DIG_B_ALT: IDDIG_B alternate status 0 = UNSET 1 = SET
11	X	ID_DIG_B: IDDIG_B status 0 = UNSET 1 = SET
10	X	ID_DIG_A_ALT: IDDIG_A alternate status 0 = UNSET 1 = SET
9	X	ID_DIG_A: IDDIG_A status 0 = UNSET 1 = SET
8	X	VDCD_DET_ALT: VDCD_DET alternate status 0 = UNSET 1 = SET
7	X	VDAT_DET_ALT: VDAT_DET alternate status 0 = UNSET 1 = SET
6	X	A_SESS_VLD_ALT: A_SESS_VLD alternate status 0 = UNSET 1 = SET
5	X	B_SESS_VLD_ALT: B_SESS_VLD alternate status 0 = UNSET 1 = SET
4	X	ID_DIG_ALT: ID alternate status 0 = UNSET 1 = SET
3	X	B_SESS_END_ALT: B_SESS_END alternate status 0 = UNSET 1 = SET

Bit	Reset	Description
2	X	STATIC_GPI_ALT: Static GPI alternate status 0 = UNSET 1 = SET
1	X	A_VBUS_VLD_ALT: A_VBUS_VLD alternate status 0 = UNSET 1 = SET
0	X	VBUS_WAKEUP_ALT: Vbus wakeup alternate status 0 = UNSET 1 = SET

23.9.2.5 USB1_IF_USB_INTER_PKT_DELAY_CTRL_0

This controls the interpacket delay between two consecutive transmit packets in HS mode. This only applies to host mode as device never transmits two packets in a row.

Inter Packet Delay Control

Offset: 0x420 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0010010

Bit	Reset	Description
6:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. Software should not change this.

23.9.2.6 USB1_IF_USB_RSM_DLY_0

USB Controller Resume Signaling Delay

Offset: 0x490 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0110100101111000

Bit	Reset	Description
15:0	0x6978	TIME_TO_RESUME: Send the resume back in number of 60 MHz cycles. Default gives 900 µs delay. Only applicable in host mode.

23.9.2.7 USB1_IF_SPARE_0

For ICUSB PADCTLS. Spare Register

Offset: 0x498 | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:16	0xffff	SPARE_HI: Spare register bits, defaulted to 1. SPARE_HI[0] - usb_port_reset_fix_en. Enable port-reset fix.
15:0	0x0	SPARE_LO: Spare register bits, defaulted to 0. SPARE_LO[4] - usb_hs_rsm_eop_fix_en, Enable HS_RESUME EOP fix. SPARE_LO[5] - usb_port_suspend_fix_en.

23.9.2.8 USB1_IF_USB1_NEW_CONTROL_0

For ULPI DIR Override

Offset: 0x4c0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx00001111xxxxxx00

Bit	Reset	Description
15:8	0xf	REQUEST_EXPIRY_COUNTER: Time to wait for coalescing the request
1	0x0	MEM_ALIGNMENT_MUX_EN: Mux to select between Tegra 3 and Tegra 4 code 0 = DISABLE 1 = ENABLE
0	0x0	COHERENCY_EN: Enable fence mechanism. 0 = DISABLE 1 = ENABLE.

23.9.3 USB1 UTMIP Configuration Registers

Note: Current UTMIP configuration registers use settings that support NV software drivers and should NOT be modified. If designing custom drivers, consult your NV representative prior to modifying these registers

23.9.3.1 UTMIP REGISTER: PLL_CFG0

This register was used to configure PLL inside UTMIP block prior to Tegra 4 devices. This has been de-featured from UTMIP space and moved to CAR space. This register is used to configure the PHY PLL contained in the UTMIP module. Refer to the Clock and Reset Controller section for details on this register.

Note: This register has been moved to the Clock and Reset section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT!

23.9.3.2 USB1_UTMIP_PLL_CFG1_0

UTMIP PLL and PLLU Configuration Register 1

Note: This register has been moved to the Clock and Reset section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT!

This register was used to configure the PLL inside the UTMIP block prior to Tegra 4 devices. This register has been defeatured from UTMIP space and moved to CAR space. Refer to the Clock and Reset Controller section for details on this register.

23.9.3.3 USB1_UTMIP_XCVR_CFG0_0

UTMIP Transceiver Cell Configuration Register 0

Offset: 0x808 | Read/Write: R/W | Reset: 0b00100000001001010110010100000000

Bit	Reset	Description
31:25	0x10	UTMIP_XCVR_HSSLEW_MSB: Most significant bits of HS_SLEW.
24:22	0x0	UTMIP_XCVR_SETUP_MSB: Most significant bits of SETUP.
21	0x1	UTMIP_XCVR_LSBIAS_SEL: Low speed bias selection method for USB transceiver pad
20	0x0	UTMIP_XCVR_DISCON_METHOD: Disconnect method on the USB transceiver pad
19	0x0	UTMIP_FORCE_PDZI_POWERUP: Force PDZI input into power up.

Bit	Reset	Description
18	0x1	UTMIP_FORCE_PDZI_POWERDOWN: Force PDZI input into power down. (Overrides FORCE_PDZI_POWERUP.)
17	0x0	UTMIP_FORCE_PD2_POWERUP: Force PD2 input into power up.
16	0x1	UTMIP_FORCE_PD2_POWERDOWN: Force PD2 input into power down. (Overrides FORCE_PD2_POWERUP.)
15	0x0	UTMIP_FORCE_PD_POWERUP: Force PD input into power up.
14	0x1	UTMIP_FORCE_PD_POWERDOWN: Force PD input into power down. (Overrides FORCE_PD_POWERUP.)
13	0x1	UTMIP_XCVR_TERMEN: Enable HS termination.
12	0x0	UTMIP_XCVR_HSLOOPBACK: Internal loopback inside XCVR cell. Used for IOBIST.
11:10	0x1	UTMIP_XCVR_LSFSLEW: LS falling slew rate control.
9:8	0x1	UTMIP_XCVR_LSRSLEW: LS rising slew rate control.
7:6	0x0	UTMIP_XCVR_FSSLEW: FS slew rate control.
5:4	0x0	UTMIP_XCVR_HSSLEW: HS slew rate control. The two LSBs.
3:0	0x0	UTMIP_XCVR_SETUP: SETUP[3:0] input of XCVR cell. HS driver output control. 4 LSBs.

23.9.3.4 USB1_UTMIP_BIAS_CFG0_0

UTMIP Bias Cell Configuration Register 0

Offset: 0x80c | Read/Write: R/W | Reset: 0bx00000001100000000000110000000000

Bit	Reset	Description
30	0x0	UTMIP_IDDIG_C_VAL: See IDDIG_C_SEL.
29	0x0	UTMIP_IDDIG_C_SEL: 0: IdDig_c = IdDig_c. 1: IdDig_c = IDDIG_C_VAL.
28	0x0	UTMIP_IDDIG_B_VAL: See IDDIG_B_SEL.
27	0x0	UTMIP_IDDIG_B_SEL: 0: IdDig_b = IdDig_b. 1: IdDig_b = IDDIG_B_VAL.
26	0x0	UTMIP_IDDIG_A_VAL: See IDDIG_A_SEL.
25	0x0	UTMIP_IDDIG_A_SEL: 0: IdDig_a = IdDig_a. 1: IdDig_a = IDDIG_A_VAL.
24	0x0	UTMIP_HSDISCON_LEVEL_MSB: Most significant bit of UTMIP_HSDISCON_LEVEL, bit 2
23	0x1	UTMIP_IDPD_VAL: See IDPD_SEL.
22	0x1	UTMIP_IDPD_SEL: 0: Reserved. Refer to the PMC registers for this feature.
21	0x0	UTMIP_IDDIG_VAL: See IDDIG_SEL.
20	0x0	UTMIP_IDDIG_SEL: 0: IdDig = IdDig. 1: IdDig = IDDIG_VAL.
19	0x0	UTMIP_GPI_VAL: See GPI_SEL.
18	0x0	UTMIP_GPI_SEL: 0: StaticGpi = IdDig. 1: StaticGpi = GPI_VAL.
17:15	0x0	UTMIP_ACTIVE_TERM_OFFSET: Active termination control offset.
14:12	0x0	UTMIP_ACTIVE_PULLUP_OFFSET: Active 1.5K pullup control offset.
11	0x1	UTMIP_OTGPD: Power down OTG circuit.
10	0x1	UTMIP_BIASPD: Power down bias circuit.
9:8	0x0	UTMIP_VBUS_LEVEL_LEVEL: Vbus detector level.
7:6	0x0	UTMIP_SESS_LEVEL_LEVEL: SessionEnd detector level.

Bit	Reset	Description
5:4	0x0	UTMIP_HSCHIRP_LEVEL: HS chirp detector level.
3:2	0x0	UTMIP_HSDISCON_LEVEL: HS disconnect detector level.
1:0	0x0	UTMIP_HSSQUELCH_LEVEL: HS squelch detector level.

23.9.3.5 USB1_UTMIP_HSRX_CFG0_0

UTMIP High Speed Receive Config 0

Offset: 0x810 | Read/Write: R/W | Reset: 0b10010001011001010011010000000000

Bit	Reset	Description
31:30	0x2	UTMIP_KEEP_PATT_ON_ACTIVE: Keep the stay alive pattern on active
29	0x0	UTMIP_ALLOW_CONSEC_UPDN: Allow consecutive ups and downs on the bits, debug only, set to 0.
28	0x1	UTMIP_REALIGN_ON_NEW_PKT: Realign the inertia counters on a new packet
27:24	0x1	UTMIP_PCOUNT_UPDN_DIV: The number of (edges-1) needed to move the sampling point
23:21	0x3	UTMIP_SQUELCH_EOP_DLY: Limit the delay of the squelch at EOP time
20	0x0	UTMIP_NO_STRIPPING: Do not strip incoming data
19:15	0xa	UTMIP_IDLE_WAIT: Number of idle cycles to declare IDLE.
14:10	0xd	UTMIP_ELASTIC_LIMIT: Depth of elastic input store
9	0x0	UTMIP_ELASTIC_OVERRUN_DISABLE: Do not declare overrun errors until overflow of FIFO
8	0x0	UTMIP_ELASTIC_UNDERRUN_DISABLE: Do not declare underrun errors
7	0x0	UTMIP_PASS_CHIRP: When in Chirp Mode, allow chirp RX data through
6	0x0	UTMIP_PASS_FEEDBACK: Pass through the feedback, do not block it.
5:4	0x0	UTMIP_PCOUNT_INERTIA: Retime the path.
3:2	0x0	UTMIP_PHASE_ADJUST: Based on incoming edges and current sampling position, adjust phase
1	0x0	UTMIP_THREE_SYNCBITS: Sync pattern detection needs 3 consecutive samples instead of 4
0	0x0	UTMIP_USE4SYNC_TRAN: Require 4 sync pattern transitions (01) instead of 3

23.9.3.6 USB1_UTMIP_HSRX_CFG1_0

UTMIP High Speed Receive Config 1

Offset: 0x814 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx010011

Bit	Reset	Description
5:1	0x9	UTMIP_HS_SYNC_START_DLY: How long to wait before start of sync launches RxActive
0	0x1	UTMIP_HS_ALLOW_KEEP_ALIVE: Allow Keep Alive packets

23.9.3.7 USB1_UTMIP_FSLSRX_CFG0_0

UTMIP Full and Low Speed Receive Config 0

Offset: 0x818 | Read/Write: R/W | Reset: 0b11111101010101001000010000101001

Bit	Reset	Description
31	0x1	UTMIP_FSLS_SE1_DRIBBLE_FILTER: One SE1, do not allow dribble

Bit	Reset	Description
30	0x1	UTMIP_FSLS_SE1_FILTER: Filter SE1
29	0x1	UTMIP_FSLS_SERIAL_SE0_RCV
28:26	0x7	UTMIP_FSLS_UPR_DRIBBLE_SIZE: Do not allow <= dribble bits
25:23	0x2	UTMIP_FSLS_LWR_DRIBBLE_SIZE: Do not allow >= dribble bits
22	0x1	UTMIP_FSLS_EOP_ENDS_AT_SE0: Only look for transitioning out of EOP
21:16	0x14	UTMIP_FSLS_KCOUNT_MAX: Number of K bits in question
15	0x1	UTMIP_FSLS_KCOUNT_LIMIT: Limit the number of bit times a K can last
14	0x0	UTMIP_FSLS_ACTIVE_ON_FULL_SYNC: Require a full sync pattern to declare the data received
13:8	0x4	UTMIP_FSLS_IDLE_WAIT_MAX: 4 bits of SEO should exceed the time limit
7	0x0	UTMIP_FSLS_IDLE_WAIT_LIMIT: Enable the reset of the state machine on extended SE0
6:1	0x14	UTMIP_FSLS_IDLE_COUNT_MAX: 20 bits of idle should end the packet if FsLsIdleCountLimitCfg=1.
0	0x1	UTMIP_FSLS_IDLE_COUNT_LIMIT: Give up on packet if a long sequence of J

23.9.3.8 USB1_UTMIP_FSLSRX_CFG1_0

UTMIP Full and Low Speed Receive Config 1

Offset: 0x81c | Read/Write: R/W | Reset: 0bxxxxx01000100110011101000000000

Bit	Reset	Description
26	0x0	UTMIP_EARLY_LINE_STATE_FILTER: Assumes line state filtering table is inclusive, not exclusive
25:23	0x4	UTMIP_LS_BOUNCE_LENGTH: Number of clock cycle of LS stable
22:17	0x13	UTMIP_LS_EXTRACTION_COUNT: Phase count on which LS bits are extracted
16:11	0xe	UTMIP_LS_EOP_START_COUNT: Number of SEO clock cycles to block bit extraction
10:5	0x20	UTMIP_LS_SE0_COUNT: Only for this number of 60MHz of SEO and Idle to end packet
4	0x0	UTMIP_LS_LENIENT_DRIBBLE: Allow for large dribble in low speed mode
3	0x0	UTMIP_FS_LENIENT_DRIBBLE: Allow for large dribble in full speed mode
2	0x0	UTMIP_FS_WEAK_SYNC: Only look for a KK pattern, instead of KJKK
1	0x0	UTMIP_FS_DEBOUNCE: Whether full speed uses debouncing
0	0x0	UTMIP_FS_EOP_LENGTH: Whether full speed EOP is determined within 3(0) or 4(1) 60MHz cycles

23.9.3.9 USB1_UTMIP_TX_CFG0_0

UTMIP Transmit Config Signals

Offset: 0x820 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxx00010000001000000000

Bit	Reset	Description
19	0x0	UTMIP_FS_PREAMBLE_J: Output enable sends an initial J before sync pattern
18	0x0	UTMIP_FS_POSTAMBLE_OUTPUT_ENABLE: Output enable turns off 1/2 cycle after
17	0x0	UTMIP_FS_PREAMBLE_OUTPUT_ENABLE: Output enable turns on 1/2 cycle before
16	0x1	UTMIP_FSLS_ALLOW_SOP_TX_STUFF_ERR: Allow SOP to be source of transmit error stuffing
15	0x0	UTMIP_HS_READY_WAIT_FOR_VALID

Bit	Reset	Description
14:10	0x0	UTMIP_HS_TX_IPG_DLY
9	0x1	UTMIP_HS_DISCON_EOP_ONLY: Only check during EOP
8	0x0	UTMIP_HS_DISCON_DISABLE: Disable high speed disconnect
7	0x0	UTMIP_HS_POSTAMBLE_OUTPUT_ENABLE: Output enable turns off 1 cycle after
6	0x0	UTMIP_HS_PREAMBLE_OUTPUT_ENABLE: Output enable turns on 1 cycle before
5	0x0	UTMIP_SIE_RESUME_ON_LINESTATE: SIE, not macrocell, detects LineState change to resume
4	0x0	UTMIP_SOF_ON_NO_STUFF: SOF when OpMode 3 -- perhaps, when sending controller made packets
3	0x0	UTMIP_SOF_ON_NO_ENCODE: SOF when OpMode 2 -- not likely, for Chirp
2	0x0	UTMIP_NO_STUFFING: No bit stuffing, static programming
1	0x0	UTMIP_NO_ENCODING: No encoding, static programming
0	0x0	UTMIP_NO_SYNC_NO_EOP: Do not send SYNC or EOP

23.9.3.10 USB1_UTMIP_MISC_CFG0_0

UTMIP Miscellaneous Configurations

Offset: 0x824 | Read/Write: R/W | Reset: 0bx00000111110000000000000001111000

Bit	Reset	Description
30:27	0x0	UTMIP_DPDM_OBSERVE_SEL: Select DP/DM obs signals
26	0x0	UTMIP_DPDM_OBSERVE: Use DP/DM as obs bus
25	0x1	UTMIP_KEEP_XCVR_PD_ON_SOFT_DISCON
24	0x1	UTMIP_ALLOW_LS_ON_SOFT_DISCON
23	0x1	UTMIP_FORCE_FS_DISABLE_ON_DEV_CHIRP
22	0x1	UTMIP_SUSPEND_EXIT_ON_EDGE: Suspend exit requires edge or simply a value.
21	0x1	UTMIP_LS_TO_FS_SKIP_4MS: Do not block changes for 4 ms when going from LS to FS (should not happen)
20:19	0x0	UTMIP_INJECT_ERROR_TYPE: Force error insertion into RX path. (Used for IOBIST.) 0 = DISABLE 1 = BIT_ERR 2 = RX_ERR 3 = BIT_RX_ERR
18	0x0	UTMIP_FORCE_HS_CLOCK_ON: Force HS clock always on.
17	0x0	UTMIP_DISABLE_HS_TERM: Force HS termination inactive.
16	0x0	UTMIP_FORCE_HS_TERM: Force HS termination active.
15	0x0	UTMIP_DISABLE_PULLUP_DP: Force DP pullup inactive. (Overrides FORCE_PULLUP_DP.)
14	0x0	UTMIP_DISABLE_PULLUP_DM: Force DM pullup inactive. (Overrides FORCE_PULLUP_DM.)
13	0x0	UTMIP_DISABLE_PULLDN_DP: Force DP pulldown inactive. (Overrides FORCE_PULLDN_DP.)
12	0x0	UTMIP_DISABLE_PULLDN_DM: Force DM pulldown inactive. (Overrides FORCE_PULLDN_DM.)
11	0x0	UTMIP_FORCE_PULLUP_DP: Force DP pullup active.
10	0x0	UTMIP_FORCE_PULLUP_DM: Force DM pullup active.
9	0x0	UTMIP_FORCE_PULLDN_DP: Force DP pulldown active.

Bit	Reset	Description
8	0x0	UTMIP_FORCE_PULLDN_DM: Force DM pulldown active.
7:5	0x3	UTMIP_STABLE_COUNT: Number of cycles of crystal clock of signal not changing to consider stable.
4	0x1	UTMIP_STABLE_ALL: Determines if all signal need to be stable to not change a config.
3	0x1	UTMIP_NO_FREE_ON_SUSPEND: Do not use free running terminations during suspend.
2	0x0	UTMIP_NEVER_FREE_RUNNING_TERMS: Ignore free-running terminations, even when no clock
1	0x0	UTMIP_ALWAYS_FREE_RUNNING_TERMS: Use free-running terminations at all time
0	0x0	UTMIP_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

23.9.3.11 USB1_UTMIP_MISC_CFG1_0

UTMIP Miscellaneous Configurations

Offset: 0x828 | Read/Write: R/W | Reset: 0bx10000000001100110000000000100100

Bit	Reset	Description
30	0x1	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
29	0x0	UTMIP_LINESTATE_BYPASS: Bypass LineState reclocking logic
28	0x0	UTMIP_LINESTATE_NEG: Use negative edge sync for line state
27	0x0	UTMIP_LINESTATE_XCVRSEL3: 0: Use FS filtering on line state when XcvrSel=3 1: Use LS filtering on line state when XcvrSel=3
26:25	0x0	UTMIP_OBS_SEL
24	0x0	UTMIP_FSLS_TDM
23	0x0	UTMIP_FORCE_IOBIST_CLK_ON
22:18	0x6	UTMIP_PLL_ACTIVE_DLY_COUNT: Reserved. Config moved to the Clock and Reset space.
17:6	0x600	UTMIP_PLLU_STABLE_COUNT: Reserved. Moved to the Clock and Reset space.
5	0x1	UTMIP_RX_ERROR_CNT_CLR
4	0x0	UTMIP_RX_ERROR_CNT_EN
3	0x0	UTMIP_FLIP_FSLS_POLARITY
2	0x1	UTMIP_SUSPEND_TERMSEL
1:0	0x0	UTMIP_XCVRSEL3: Bit 0: 0: 0xa5 -> treat as KeepAlive 1: treat as regular packet Bit 1: 0: Turn on FS EOP detection 1: Turn off FS EOP detection

23.9.3.12 USB1_UTMIP_DEBOUNCE_CFG0_0

UTMIP Avalid and Bvalid Debounce

Debounce values IdDig, Avalid, Bvalid, VbusValid, VbusWakeUp, and SessEnd. Each of these signals has its own debouncer, and for each of those one out of 2 debouncing times can be chosen (BIAS_DEBOUNCE_A or BIAS_DEBOUNCE_B.)

The values of DEBOUNCE_A and DEBOUNCE_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2MHz) / 4

So to program a 1 ms debounce for BIAS_DEBOUNCE_A:

BIAS_DEBOUNCE_A[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 0x82c | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_BIAS_DEBOUNCE_B: simulation value -- Used for interrupts
15:0	0xffff	UTMIP_BIAS_DEBOUNCE_A: simulation value -- Used for interrupts

23.9.3.13 USB1_UTMIP_BAT_CHRG_CFG0_0

UTMIP Battery Charger Configuration

Offset: 0x830 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxx000000xx000001

Bit	Reset	Description
13:8	0x0	UTMIP_CHRG_DEBOUNCE_TIMESCALE: Debouncer time scaling, factor-1 to slow down debouncing by. So 0 is 1, 1 is 2, etc.
5	0x0	UTMIP_OP_I_SRC_EN
4	0x0	UTMIP_ON_SRC_EN
3	0x0	UTMIP_OP_SRC_EN
2	0x0	UTMIP_ON_SINK_EN
1	0x0	UTMIP_OP_SINK_EN
0	0x1	UTMIP_PD_CHRG: Power down charger circuit

23.9.3.14 USB1_UTMIP_SPARE_CFG0_0

UTMIP Spare Configuration Bits

Offset: 0x834 | Read/Write: R/W | Reset: 0b11111111111111110000000011111000

Bit	Reset	Description
31:0	-65288	UTMIP_SPARE: Spare register bits 0: HS_RX_IPG_ERROR_ENABLE 1: HS_RX_FLUSH_ALAP. Flush as late as possible 2: HS_RX_LATE_SQUELCH. Delay Squelch by 1 CLK480 cycle. 3: FUSE_SETUP_SEL. Select between regular CFG value and JTAG values for UX_SETUP 4: FUSE_TERM_RANGE_ADJ_SEL. Select between regular CFG value and JTAG values for UX_TERM_RANGE_ADJ 5: FUSE_SPARE. Select between regular CFG value and JTAG values. For any ECOs. 6: FUSE_HS_SQUELCH_LEVEL. Select between regular CFG value and JTAG values 7: FUSE_HS_IREF_CAP_CFG. Select between regular CFG value and JTAG values 31 to 8: Reserved

23.9.3.15 USB1_UTMIP_XCVR_CFG1_0

UTMIP Transceiver Cell Configuration Register 1

Offset: 0x838 | Read/Write: R/W | Reset: 0bxxxx00000001100010000001000010101

Bit	Reset	Description
27:26	0x0	UTMIP_XCVR_RPU_RANGE_ADJ: 1.5k pull-up resistor range shift.
25:24	0x0	UTMIP_XCVR_HS_IREF_CAP: High-speed Iref cap control for bias current stability.
23:22	0x0	UTMIP_XCVR_SPARE: Spare bits for USB transceiver pad.

Bit	Reset	Description
21:18	0x6	UTMIP_XCVR_TERM_RANGE_ADJ: Range adjustment on terminations.
17	0x0	UTMIP_RCTRL_SW_SET: Use a software override on RCTRL instead of automatic bias control.
16:12	0x8	UTMIP_RCTRL_SW_VAL: Encoded value to use on RCTRL when software override is enabled, 0 to 16 only.
11	0x0	UTMIP_TCTRL_SW_SET: Use a software override on TCTRL instead of automatic bias control
10:6	0x8	UTMIP_TCTRL_SW_VAL: Encoded value to use on TCTRL when software override is enabled, 0 to 16 only.
5	0x0	UTMIP_FORCE_PDDR_POWERUP: Force PDDR input into power up.
4	0x1	UTMIP_FORCE_PDDR_POWERDOWN: Force PDDR input into power down. (Overrides FORCE_PDDR_POWERUP.)
3	0x0	UTMIP_FORCE_PDCHRP_POWERUP: Force PDCHRP input into power up.
2	0x1	UTMIP_FORCE_PDCHRP_POWERDOWN: Force PDCHRP input into power down. (Overrides FORCE_PDCHRP_POWERUP.)
1	0x0	UTMIP_FORCE_PDDISC_POWERUP: Force PDDISC input into power up.
0	0x1	UTMIP_FORCE_PDDISC_POWERDOWN: Force PDDISC input into power down. (Overrides FORCE_PDDISC_POWERUP.)

23.9.3.16 USB1_UTMIP_BIAS_CFG1_0

UTMIP Bias Cell Configuration Register 1

Offset: 0x83c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxx00000000101101

Bit	Reset	Description
13:8	0x0	UTMIP_BIAS_DEBOUNCE_TIMESCALE: Debouncer time scaling. Slows down debouncing by a factor-1. So 0 is 1, 1 is 2, etc.
7:3	0x5	UTMIP_BIAS_PDTRK_COUNT: Control the BIAS cell power down lag. The lag should be 20 µs. For a crystal clock of 13 MHz, it should be set to 5.
2	0x1	UTMIP_VBUS_WAKEUP_POWERDOWN: Reserved. See the PMC registers for this functionality.
1	0x0	UTMIP_FORCE_PDTRK_POWERUP: Force PDTRK input into power up.
0	0x1	UTMIP_FORCE_PDTRK_POWERDOWN: Force PDTRK input into power down. (Overrides FORCE_PDTRK_POWERUP.)

23.9.3.17 USB1_UTMIP_BIAS_STS0_0

UTMIP Bias Cell Status Register 0

Offset: 0x840 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	UTMIP_TCTRL: Thermal encoding output from USB bias pad.
15:0	X	UTMIP_RCTRL: Thermal encoding output from USB bias pad.

23.9.3.18 USB1_UTMIP_CHRG_DEB_CFG0_0

UTMIP VDcd_Det and VDat_Det Debounce

Debounce values VDcd_Det and VDat_Det. Each of these signals has its own debouncer and for each of those, 1 out of 2 debouncing times can be chosen (CHRG_DEBOUNCE_PERIOD_A or CHRG_DEBOUNCE_PERIOD_B).

The values of DEBOUNCE_PERIOD_A and DEBOUNCE_PERIOD_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2MHz) / 4

So to program a 1 ms debounce for CHG_DEBOUNCE_PERIOD:

CHG_DEBOUNCE_PERIOD[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 0x844 | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_CHRG_DEBOUNCE_PERIOD_B: simulation value -- Used for interrupts
15:0	0xffff	UTMIP_CHRG_DEBOUNCE_PERIOD_A: simulation value -- Used for interrupts

23.9.3.19 USB1_UTMIP_MISC_STS0_0

UTMIP MISC Status Register

Offset: 0x848 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
21	X	UTMIP_FS_DRV_EN: Indicates when the controller is driving on the bus
20:0	X	UTMIP_SPARE_FUSES: Spare Fuses value to keep the connections preserved

23.9.3.20 USB1_UTMIP_PMC_WAKEUP0_0

UTMIP PMC Wakeup value

Offset: 0x84c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
1	RO	X	UTMIP_LINE_WAKEUP_EVENT_INT_STS: Indicates the status of the PMC wakeup event
0	RW	0x0	UTMIP_LINE_WAKEUP_EVENT_INT_ENB: Interrupt mask for the PMC wakeup event

23.9.3.21 USB2_QH_USB2D_QH_EP_0_OUT_0

USB2D Queue Head for OUT Endpoint 0

Offset: 0x1000 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 0. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 0.

23.9.3.22 USB2_QH_USB2D_QH_EP_0_IN_0

USB2D Queue Head for IN Endpoint 0

Offset: 0x1040 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 0. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 0.

23.9.3.23 USB2_QH_USB2D_QH_EP_1_OUT_0

USB2D Queue Head for OUT Endpoint 1

Offset: 0x1080 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 1. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 1.

23.9.3.24 USB2_QH_USB2D_QH_EP_1_IN_0

USB2D Queue Head for IN Endpoint 1

Offset: 0x10c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 1. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 1.

23.9.3.25 USB2_QH_USB2D_QH_EP_2_OUT_0

USB2D Queue Head for OUT Endpoint 2

Offset: 0x1100 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 2. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 2.

23.9.3.26 USB2_QH_USB2D_QH_EP_2_IN_0

USB2D Queue Head for IN Endpoint 2

Offset: 0x1140 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 2. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 2.

23.9.3.27 USB2_QH_USB2D_QH_EP_3_OUT_0

USB2D Queue Head for OUT Endpoint 3

Offset: 0x1180 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 3. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 3.

23.9.3.28 USB2_QH_USB2D_QH_EP_3_IN_0

USB2D Queue Head for IN Endpoint 3

Offset: 0x11c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 3. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 3.

23.9.3.29 USB2_QH_USB2D_QH_EP_4_OUT_0

USB2D Queue Head for OUT Endpoint 4

Offset: 0x1200 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 4. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 4.

23.9.3.30 USB2_QH_USB2D_QH_EP_4_IN_0

USB2D Queue Head for IN Endpoint 4

Offset: 0x1240 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 4. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 4.

23.9.3.31 USB2_QH_USB2D_QH_EP_5_OUT_0

USB2D Queue Head for OUT Endpoint 5

Offset: 0x1280 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 5. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 5.

23.9.3.32 USB2_QH_USB2D_QH_EP_5_IN_0

USB2D Queue Head for IN Endpoint 5

Offset: 0x12c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 5. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 5.

23.9.3.33 USB2_QH_USB2D_QH_EP_6_OUT_0

USB2D Queue Head for OUT Endpoint 6

Offset: 0x1300 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 6. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 6.

23.9.3.34 USB2_QH_USB2D_QH_EP_6_IN_0

USB2D Queue Head for IN Endpoint 6

Offset: 0x1340 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 6. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 6.

23.9.3.35 USB2_QH_USB2D_QH_EP_7_OUT_0

USB2D Queue Head for OUT Endpoint 7

Offset: 0x1380 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 7. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 7.

23.9.3.36 USB2_QH_USB2D_QH_EP_7_IN_0

USB2D Queue Head for IN Endpoint 7

Offset: 0x13c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 7. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 7.

23.9.3.37 USB2_QH_USB2D_QH_EP_8_OUT_0

USB2D Queue Head for OUT Endpoint 8

Offset: 0x1400 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 8. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 8.

23.9.3.38 USB2_QH_USB2D_QH_EP_8_IN_0

USB2D Queue Head for IN Endpoint 8

Offset: 0x1440 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 8. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 8.

23.9.3.39 USB2_QH_USB2D_QH_EP_9_OUT_0

USB2D Queue Head for OUT Endpoint 9

Offset: 0x1480 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 9. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 9.

23.9.3.40 USB2_QH_USB2D_QH_EP_9_IN_0

USB2D Queue Head for IN Endpoint 9

Offset: 0x14c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 9. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 9.

23.9.3.41 USB2_QH_USB2D_QH_EP_10_OUT_0

USB2D Queue Head for OUT Endpoint 10

Offset: 0x1500 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 10. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 10.

23.9.3.42 USB2_QH_USB2D_QH_EP_10_IN_0

USB2D Queue Head for IN Endpoint 10

Offset: 0x1540 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 10. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 10.

23.9.3.43 USB2_QH_USB2D_QH_EP_11_OUT_0

USB2D Queue Head for OUT Endpoint 11

Offset: 0x1580 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 11. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 11.

23.9.3.44 USB2_QH_USB2D_QH_EP_11_IN_0

USB2D Queue Head for IN Endpoint 11

Offset: 0x15c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 11. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 11.

23.9.3.45 USB2_QH_USB2D_QH_EP_12_OUT_0

USB2D Queue Head for OUT Endpoint 12

Offset: 0x1600 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 12. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 12.

23.9.3.46 USB2_QH_USB2D_QH_EP_12_IN_0

USB2D Queue Head for IN Endpoint 12

Offset: 0x1640 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 12. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 12.

23.9.3.47 USB2_QH_USB2D_QH_EP_13_OUT_0

USB2D Queue Head for OUT Endpoint 13

Offset: 0x1680 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 13. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 13.

23.9.3.48 USB2_QH_USB2D_QH_EP_13_IN_0

USB2D Queue Head for IN Endpoint 13

Offset: 0x16c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 13. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 13.

23.9.3.49 USB2_QH_USB2D_QH_EP_14_OUT_0

USB2D Queue Head for OUT Endpoint 14

Offset: 0x1700 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 14. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 14.

23.9.3.50 USB2_QH_USB2D_QH_EP_14_IN_0

USB2D Queue Head for IN Endpoint 14

Offset: 0x1740 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 14. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 14.

23.9.3.51 USB2_QH_USB2D_QH_EP_15_OUT_0

USB2D Queue Head for OUT Endpoint 15

Offset: 0x1780 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 15. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 15.

23.9.3.52 USB2_QH_USB2D_QH_EP_15_IN_0

USB2D Queue Head for IN Endpoint 15

Offset: 0x17c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 15. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 15.

23.9.4 USB2 Controller Registers

23.9.4.1 USB2_CONTROLLER_1_USB2D_ID_0

USB2D Identification Register

Offset: 0x0 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	CIVERSION: Identifies the CI version
28:25	X	VERSION: Identifies the version of the core
24:21	X	REVISION: Revision number of the USB controller. This is set to 0x0.
20:16	X	TAG: Identifies the tag of the core
15:8	X	NID: One's complement version of ID. This field is set to 0xF9.
7:0	X	ID: Configuration number. This field is set to 0x06

23.9.4.2 USB2_CONTROLLER_1_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 0x8 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
3:1	X	NPORT: VUSB_HS_NUM_PORT-1: This host controller has only 1 port. So this field will always be 0.
0	X	HC: VUSB_HS_HOST: Indicates support for host mode. Set to 1.

23.9.4.3 USB2_CONTROLLER_1_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

Offset: 0xc |Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: Number of endpoints supported by this device controller. Set to 16. This includes control endpoint 0.
0	X	DC: Device capable: Set to 1 indicating support for device mode.

23.9.4.4 USB2_CONTROLLER_1_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 0x10 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total number of address bits for the transmit buffer of each transmit endpoint. Set to 7. Each transmit buffer is 128 words deep.
15:8	X	TXADD: VUSB_HS_TX_ADD: Total number of address bits for the transmit buffer. Set to 11. The total depth of the transmit buffer is 2048 words.
7:0	X	TCBURST: VUSB_HS_TX_BURST: Maximum burst size supported by the transmit endpoints for data transfers. Set to 8.

23.9.4.5 USB2_CONTROLLER_1_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 0x14 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total number of address bits for the receive buffer. Set to 7. The total depth of the receive buffer is 128 words.
7:0	X	RXBURST: VUSB_HS_RX_BURST: Maximum burst size supported by the receive endpoints for data transfers. Set to 8.

23.9.4.6 USB2_CONTROLLER_1_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

Offset: 0x100 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
7:0	X	CAPLENGTH: Indicates which offset to add to the register base address at the beginning of the Operational Register. Set to 0x40.

23.9.4.7 USB2_CONTROLLER_1_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 0x102 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	HCIVERSION: Contains a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. This host controller supports EHCI revision 1.00.

23.9.4.8 USB2_CONTROLLER_1_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

Offset: 0x104 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
27:24	X	N_TT: Number of Transaction Translators: Indicates the number of embedded transaction translators associated with the USB2.0 host controller. This field is always set to 1 indicating only 1 embedded TT is implemented in this implementation. This is a non-EHCI field to support embedded TT.
23:20	X	N_PTT: Number of Ports per Transaction Translator: Indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. Field always equals N_PORTS. This is a non-EHCI field to support embedded TT.
15:12	X	N_CC: Number of Companion Controller: Indicates the number of companion controllers. This field is set to 0.
11:8	X	N_PCC: Number of Ports per Companion Controller: Indicates the number of ports supported per internal companion controller. This field is set to 0.
4	X	PPC: Port Power Control: Indicates whether the host controller implementation includes port power control. 1 = Ports have port power switches 0= Ports do not have port power switches. This field affects the functionality of the port Power field in each port status and control register. This field is set to 1.
3:0	X	N_PORTS: Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. This field is fixed to 1, since this host controller only supports 1 port.

23.9.4.9 USB2_CONTROLLER_1_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

Offset: 0x108 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18	X	PPC: Per-Port Change Event Capability. Default = 1b. This field indicates the support for per-port change events. This field is related to the USBCMD PPE field, USBSTS PPCI field, and USBINTR PPCE field.
17	X	LEN: Link Power Management Capability. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the USBCMD HIRD field, POSTSCx SSTS and DA fields, and HOSTPCx LEN, BA, and EPLPM fields.
15:8	X	EECP: EHCI Extended Capabilities Pointer: Indicates a capabilities list exists. A value of 00h indicates no extended capabilities are implemented. For this implementation this field is always "0".
7:4	X	IST: Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be "0".
2	X	ASP: Asynchronous Schedule Park Capability. 1 = (Default) the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register. This field is always 1.
1	X	PFL: Programmable Frame List Flag. 0 = System software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. 1 = System software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This field will always be "1".

23.9.4.10 USB2_CONTROLLER_1_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

Offset: 0x120 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	DCIVERSION: The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

23.9.4.11 USB2_CONTROLLER_1_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 0x124 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
8	X	HC: Host Capable: 1 = This controller is capable of operating as an EHCI compatible USB 2 0 host controller operating as an EHCI compatible USB 2.0 host controller. This field is set to 1.
7	X	DC: Device Capable: 1 = Controller is capable of operating as USB 2.0 device. This field is set to 1
5	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the DEVLCx ASUS, STL,BA and NYT fields.
4:0	X	DEN: Device Endpoint Number: Number of endpoints built into the device controller. This is set to 16.

23.9.4.12 USB2_CONTROLLER_1_USB2D_EXTSTS_0

USB2D EXTSTS Register

Offset: 0x128 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000x

Bit	R/W	Reset	Description
4	RW	0x0	TI1: General Purpose Timer Interrupt 1 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
3	RW	0x0	TI0: General Purpose Timer Interrupt 0 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
2	RW	0x0	UPA: USB Host Periodic Interrupt (USBHSTPERINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
1	RW	0x0	UAI: USB Host Asynchronous Interrupt (USBHSTASYNCINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
0	RO	X	NAKI: NAK Interrupt Bit Read Only. This bit is read only. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared. 0 = DISABLE 1 = ENABLE

23.9.4.13 USB2_CONTROLLER_1_USB2D_USBEXTINTR_0

USB2D EXTINTR Register

Offset: 0x12c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000

Bit	Reset	Description
4	0x0	TIE1: General Purpose Timer Interrupt Enable 1 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
3	0x0	TIE0: General Purpose Timer Interrupt Enable 0 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
2	0x0	UPIE: UPIE Interrupt Enable. 1 = USB controller issues an interrupt if the UPA bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
1	0x0	UAIE: UAIE Interrupt Enable. 1 = USB controller issues an interrupt if the UAI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
0	0x0	NAKE: NAK Interrupt Enable. 1 = USB controller issues an interrupt if the NAKI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE

23.9.4.14 USB2_CONTROLLER_1_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 0x130 | Read/Write: R/W | Reset: 0bxxxx000000001000000x1x11x0000000

Bit	R/W	Reset	Description
27:24	RW	0x0	HIRD: Host Initiated Resume Duration RW. Default = 0000b. This has the same behavior as bits 7:4 of the BA field of the HOSTPCx register. When writing to this field all BA[7:4] fields of all HOSTPCx registers will be set to this value. This field is used by system software to specify the minimum amount of time the host controller will drive the K-state during a host-initiated resume from an LPM state (e.g., L1), and is conveyed to each LPM-enabled device (via the HIRD bits within an LPM Tokens bmAttributes field) upon entry into a low-power state. Note the host controller is required to drive resume signaling for at least the amount of time specified in the HIRD value conveyed to the device during any proceeding host-initiated resume. Also note that the host controller is not required to observe this requirement during device-initiated resumes. Encoding for this field is identical to the definition for the similarly named HIRD field within an LPM Token, specifically: a value 0000b equals 50µs and each additional increment adds 75µs. For example, the value 0001b equals 125µs, and a value 1111b equals 1,175µs (~1.2ms).
23:16	RW	0x8	ITC: Interrupt Threshold Control .Read/Write. Default 08h. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 00h Immediate (no threshold) 01h 1 micro-frame 02h 2 micro-frames 04h 4 micro-frames 08h 8 micro-frames 10h 16 micro-frames 20h 32 micro-frames 40h 64 micro-frames 0 = IMMEDIATE 2 = ONE_MF

Bit	R/W	Reset	Description
			4 = TWO_MF 8 = EIGHT_MF 16 = SIXTEEN_MF 32 = THIRTY_TWO_MF 64 = SIXTY_FOUR_MF
15	RW	0x0	FS2: Bit 2 of Frame List Size.
14	RW	0x0	ATDTW: Add DTD Tripwire. This bit is used as a semaphore when a dTD is added to an active (primed) endpoint. This bit is set and cleared by software and will be cleared by hardware when a hazard exists such that adding a dTD to a primed endpoint may go unnoticed. 0 = CLEAR 1 = SET
13	RW	0x0	SUTW: Setup Tripwire. This bit is used as a semaphore when the 8 bytes of setup data read extracted by the firmware. If the setup lockout mode is off, then there exists a hazard when new setup data arrives and firmware is copying setup data from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists. 0 = CLEAR 1 = SET
11	RW	0x1	ASPE: Asynchronous Schedule Park mode Enable. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. This field is set to "1" in this implementation. 0 = DISABLE 1 = ENABLE
9:8	RW	0x3	ASP1_ASP0: Asynchronous Schedule Park Mode Count (OPTIONAL) Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is RO. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior. This field is set to 3h in this implementation and is Read/Write capable.
7	RO	X	LR: Light Host/Device Controller Reset (OPTIONAL) . Read Only. Not Implemented. This field will always be "0".
6	RW	0x0	IAA: Interrupt on Async Advance Doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results. 0 = CLEAR 1 = SET
5	RW	0x0	ASE: Asynchronous Schedule Enable. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
4	RW	0x0	PSE: Periodic Schedule Enable. This bit controls whether the host controller skips processing the Periodic Schedule. 0 = Do not process the Periodic Schedule 1 = Use the PERIODICLISTBASE register to access the Periodic Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
3:2	RW	0x0	FS1_FS0: Frame List Size. (Read/Write). 000 = Default. This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. Hence this field is Read/Write for this implementation. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3, and 2. 000 = 1024 elements (4096 bytes) Default value 001 = 512 elements (2048 bytes)

Bit	R/W	Reset	Description
			010 = 256 elements (1024 bytes) 011 = 128 elements (512 bytes) 100 = 64 elements (256 bytes) 101 = 32 elements (128 bytes) 110 = 16 elements (64 bytes) 111 = 8 elements (32 bytes) Only the host controller uses this field.
1	RW	0x0	RST: Controller Reset. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host Controller: When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller results in undefined behavior. Device Controller: When software writes a one to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines, etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. Writing a one to this bit in device mode is not recommended. 0 = CLEAR 1 = SET
0	RW	0x0	RS: Run/Stop: Host Controller: When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HCHalted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the host controller is in the Halted state (i.e., HCHalted in the USBSTS register is a one). Device Controller: Writing a one to this bit will cause the device controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this will cause a detach event. 0 = STOP 1 = RUN

23.9.4.15 USB2_CONTROLLER_1_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 0x134 | Read/Write: R/W | Reset: 0b00000000000000000x100x0000x0000

Bit	R/W	Reset	Description
31:16	RW	0x0	PPCI: Port-n Change Detect - RW. Default = 0000h. The definition for each bit is identical to the Port Change Detect field (bit 2 of this register) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set to a one then a port change event was detected on Port 2. The N_PORTS field in HCSPARAMS specifies how many ports are exposed by the host controller and thus how many bits in this field are valid.
15	RW	0x0	AS: Asynchronous Schedule Status. This bit reports the current status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. If AS = ASE: 1= Enable Asynchronous Schedule. 0= Disable Asynchronous Schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
14	RW	0x0	PS: Periodic Schedule Status. This bit reports the current status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. If PS = PSE:: 1 = Periodic Schedule is enabled. 0 = Periodic Schedule is disabled. Only used by the host controller. 0 = DISABLE 1 = ENABLE
13	RO	X	RCL: Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE
12	RW	0x1	HCH: HCHalted. 1 = Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g., internal error). Only used by the host controller. 0 = UNHALTED 1 = HALTED
11	RW	0x0	UALT_INT: ULPI alt_int Interrupt. 0 = Default. This interrupt bit is set when an RXCMD is received through the ULPI interface with bit 7 set (alt_int). The alt_int bit is set when an unmasked event occurs on any bit in the Carkit Interrupt Latch Register, in the ULPI PHY. The software should read the Carkit Interrupt Latch Register (Read to Clear) through the ULPI Viewport to check the source of the interrupt. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1. 0 = NOT_ULPI_ALT_INT 1 = ULPI_ALT_INT
10	RW	0x0	ULPI_INT: ULPI Interrupt. This bit is set whenever an interrupt is received from ULPI PHY. Software writes 1 to clear it. 0 = NOT_ULPI_INT 1 = ULPI_INT
8	RW	0x0	SLI: DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller. 0 = NOTSUSPEND 1 = SUSPENDED
7	RW	0x0	SRI: SOF Received. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When an SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125 µs in HS mode and will be synchronized to the actual SOF that is received. Since device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to the connect and chirp. In host mode, this bit will be set every 125 µs and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit. 0 = SOF_NOT_RCVD 1 = SOF_RCVD
6	RW	0x0	URI: USB Reset Received. When the device controller detects a USB Reset and enters the default state, this bit is set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. 0 = NO_USB_RESET 1 = USB_RESET
5	RW	0x0	AAI: Interrupt and Asynchronous Advance. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller. 0 = NOT_ADVANCED 1 = ADVANCED
4	RO	X	SEI: System Error. This bit is not used in this implementation and will always be set to "0". 0 = NO_ERROR 1 = ERROR

Bit	R/W	Reset	Description
3	RW	0x0	FRI: Frame List Rollover. The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [1 3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FHINDEX [12] toggles. Only used by the host controller. 0 = NO_ROLLOVER 1 = ROLLOVER
2	RW	0x0	PCI: Port Change Detect. The Host Controller sets this bit to a 1 when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operational states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively. This bit is not EHCI compatible. 0 = NO_PORT_CHANGE 1 = PORT_CHANGE
1	RW	0x0	UEI: USB Error Interrupt. This bit gets set by the Host/Device controller when completion of a USB transaction results in an error condition. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also ad its interrupt on complete (IOC) bit set. 0 = NO_ERROR 1 = ERROR
0	RW	0x0	UI: USB Interrupt. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) as an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes. 0 = NO_INT 1 = INT

23.9.4.16 USB2_CONTROLLER_1_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 0x138 | Read/Write: R/W | Reset: 0b0000000000000000xxxx00x000000000

Bit	Reset	Description
31:16	0x0	PPCE: Port-n Change Detect Enable - RW. Default = 0000h. The definition for each bit in this field is identical to bit 2 of this register (Port Change Interrupt Enable) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set (1b) then a port change event was detected on Port 2. When a bit in this field is a one, and the corresponding Port-n Change Detect bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port-n Change Detect bit.
11	0x0	UALTIE: ULPI alt_int Interrupt Enable. 1 = USB controller issues an interrupt if the ULPI_ALT_INT bit in the USBSTS register transitions. The interrupt is acknowledged by software writing a 1 to the ULPI_ALT_INT bit. 0 = DISABLE 1 = ENABLE
10	0x0	ULPIE: ULPI Interrupt Enable. 1 = USB controller issues an interrupt if ULPI_INT bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_INT bit. 0 = DISABLE 1 = ENABLE
8	0x0	SLE: Sleep Enable. 1 = Device controller issues an interrupt if DCSuspend bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the DCSuspend bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x0	SRE: SOF Received Enable. 1 = Device controller issues an interrupt if SOF Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the SOF Received bit. 0 = DISABLE 1 = ENABLE
6	0x0	URE: USB Reset Enable.1 = Device controller issues an interrupt if USB Reset Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
5	0x0	AAE: Interrupt on Asynchronous Advance Enable. 1 = the host controller issues an interrupt at the next interrupt threshold if the Interrupt on Async Advance bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
4	0x0	SEE: System Error Enable. 1 = Host/device controller issues an interrupt if the System Error bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the System Error bit. 0 = DISABLE 1 = ENABLE
3	0x0	FRE: Frame List Rollover Enable. 1 = Host controller issues an interrupt if the Frame List Rollover bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
2	0x0	PCE: Port Change Detect Enable. 1 = Host/device controller issues an interrupt if Port Change Detect bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Port Change Detect bit. 0 = DISABLE 1 = ENABLE
1	0x0	UEE: USB Error Interrupt Enable. 1 = Host controller issues an interrupt at the next interrupt threshold if the USBERRINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register. 0 = DISABLE 1 = ENABLE
0	0x0	UE: USB Interrupt Enable. 1 = Host/device issues an interrupt at the next interrupt threshold if the USBINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBINT bit. 0 = DISABLE 1 = ENABLE

23.9.4.17 USB2_CONTROLLER_1_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 0x13c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
13:0	X	FRINDEX: Frame Index. The value in this register increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. Each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N 000b (1024) 12 001b (512) 11 010b (256) 10 011b (128) 9 100b (64) 8 101b (32) 7 110b (16) 6 111b (8) 5 In device mode, the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode, bits 2:0 indicate the current micro-frame.

23.9.4.18 USB2_CONTROLLER_1_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 0x144 | Read/Write: R/W | Reset: 0b00000000000000000000xxxxxxxxxxxx

Bit	Reset	Description
31:25	0x0	USBADR: Device mode. The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS request.
24	0x0	USBADRA: Device Address Advance. Default=0. When this bit is 0, any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD cannot write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
31:12	0x0	BASEADR: Host mode: This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.

23.9.4.19 USB2_CONTROLLER_1_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 0x148 | Read/Write: R/W | Reset: 0b00000000000000000000000000xxxxx

Bit	Reset	Description
31:11	0x0	EPBASE: Device mode. This register contains the address of the top of the endpoint list in system memory. These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). Only used by the device controller.
31:5	0x0	ASYBASE: Host mode. This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.

23.9.4.20 USB2_CONTROLLER_1_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 0x14c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0x

Bit	R/W	Reset	Description
1	RW	0x0	TTAC: Embedded TT Async Buffers Clear. (Read/Write to set). This field will clear all pending transactions in the embedded TT Async Buffer(s). The clear will take as much time as necessary to clear buffer without interfering with a transaction in progress. TTAC will return to zero after being set by software only after the actual clear occurs.
0	RO	X	TTAS: Embedded TT Async Buffers Status. (Read Only). This read-only bit will be 1 if one or more transactions are being held in the embedded TT Async. Buffers. When this bit is a zero, then all outstanding transactions in the embedded TT have been flushed.

23.9.4.21 USB2_CONTROLLER_1_USB2D_BURSTSIZE_0

USB2D Burst Size register

Offset: 0x150 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000100000001000

Bit	Reset	Description
15:8	0x8	TXPBURST: Programmable TX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.
7:0	0x8	RXPBURST: Programmable RX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory.

23.9.4.22 USB2_CONTROLLER_1_USB2D_TXFILLTUNING_0

USB2D Transmit Fill Tuning Register

Offset: 0x154 | Read/Write: R/W | Reset: 0bxxxxxxxxxx000010xxx0000000000000

Bit	Reset	Description
21:16	0x2	TXFIFOTHRES: FIFO Burst Threshold. (Read/Write). This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2, and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.

Bit	Reset	Description
12:8	0x0	TXSCHHEALTH: Scheduler Health Counter. (Read/Write To Clear) [Default = 0] This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max out at 31.
7:0	0x0	TXSCHOH: Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267 µs when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333 µs when a device is connected in Low/Full Speed Mode

23.9.4.23 USB2_CONTROLLER_1_USB2D_ICUSB_CTRL_0

USB2D ICUSB Control Register

This register enables and controls the ICUSB FS/LS transceiver.

Offset: 0x15c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000

Bit	Reset	Description
3	0x0	IC_ENB1: ICUSB transceiver enable. This bit enables the ICUSB transceiver. To enable the interface, the bits PTS must be set to 11 in the PORTSCx. Writing a '1' to this bit selects the IC_USB interface. 0 = DISABLE 1 = ENABLE
2:0	0x0	IC_VDD1: ICUSB voltage select. It selects which voltage is being supplied to the ICUSB peripheral. 000= No voltage 001 = 1.0V - reserved 010= 1.2V - reserved 011= 1.5V - reserved 100 = 1.8V 101 = 3.0V 110 = reserved 111 = reserved The Voltage negotiation should happen between enabling port power (PP) and asserting the run/stop bit in register.

23.9.4.24 USB2_CONTROLLER_1_USB2D_ULPI_VIEWPORT_0

This register provides indirect access to the ULPI PHY register set. Although the USB controller performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.

Note: WRITES TO THE ULPI THROUGH THE VIEWPORT CAN SUBSTANTIALLY HARM STANDARD USB OPERATIONS. CURRENTLY NO USAGE MODEL HAS BEEN DEFINED WHERE SOFTWARE SHOULD NEED TO EXECUTE WRITES DIRECTLY TO THE ULPI PHY. SEE EXCEPTION REGARDING OPTIONAL FEATURES BELOW.

Note: EXECUTING READ OPERATIONS THROUGH THE ULPI VIEWPORT SHOULD HAVE NO HARMFUL SIDE EFFECTS TO STANDARD USB OPERATIONS.

There are two operations that can be performed with the ULPI Viewport-- wakeup and read /write operations.

The wakeup operation is used to put the ULPI interface into normal operation mode and re-enable the clock if necessary. A wakeup operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or Carkit mode.

The ULPI state can be determined by reading the sync state bit (ULPI_SYNC_STATE). If this bit is a one, then ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPI_SYNC_STATE indicates a 0 then then

read/write operations will not be able to execute. Undefined behavior will result if ULPI_SYNC_STATE = 0 and a read or write operation is performed.

To execute a wakeup operation, write all 32 bits of the ULPI Viewport where ULPI_PORT is constructed appropriately and the ULPI_WAKEUP bit is a 1 and ULPI_RUN bit is a 0. Poll the ULPI Viewport until ULPI_WAKEUP is zero for the operation to complete.

To execute a read or write operation, write all 32 bits of the ULPI Viewport where ULPI_DATA_WR, ULPI_REG_ADDR, ULPI_PORT, ULPI_RD_WR are constructed appropriately and the ULPI_RUN bit is a 1. Poll the ULPI Viewport until ULPI_RUN is zero for the operation to complete. Once ULPI_RUN is zero, the ULPI_DATA_RD will be valid if the operation was a read.

The polling method above can be changed to interrupt driven using the ULPI interrupt defined in the USBSTS and USBINTR registers. When a wakeup or read/write operation is complete, the ULPI_INT interrupt will be set.

There are several optional features that may need to be enabled or disabled by system software as part of system configuration. These bits are contained in the Interface and OTG Control registers of the ULPI PHY register set. These registers also contain bits which are controlled by the link dynamically and therefore should be only modified by system software using the Set/Clear access method. Direct writes to these registers could have harmful side effects to the standard USB operations. The optional bits are as follows: Bits 3 through 7 in the Interface Control register and Bits 6 and 7 in the OTG Control register.

Please refer to the ULPI Specification Revision 1.1 for further information on the use of the optional features.

USB2D ULPI Viewport Register

Offset: 0x160 | Read/Write: R/W | Reset: 0b000xx00000000000xxxxxxxx00000000

Bit	R/W	Reset	Description
31	RW	0x0	ULPI_WAKEUP: ULPI Wakeup. Writing the 1 to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
30	RW	0x0	ULPI_RUN: ULPI read/write Run. Writing the 1 to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
29	RW	0x0	ULPI_RD_WR: ULPI read/write control. (0) Read; (1) Write. This bit selects between running a read or write operation. 0 = READ 1 = WRITE
27	RO	X	ULPI_SYNC_STATE: ULPI sync state. (1) Normal Sync. State. (0) In another state (i.e., Carkit, serial, low power). This bit represents the state of the ULPI interface. 0 = NOT_NORMAL 1 = NORMAL
26:24	RW	0x0	ULPI_PORT: ULPI PHY port number. This field should be always written as 0.
23:16	RW	0x0	ULPI_REG_ADDR: ULPI PHY register address. When doing a read or write operation to the ULPI PHY, the address of the ULPI PHY register being accessed is written to this field.
15:8	RO	X	ULPI_DATA_RD: ULPI PHY data read. The data from the ULPI PHY register can be read from here after the read operation completes.
7:0	RW	0x0	ULPI_DATA_WR: ULPI PHY data write. The data to write to the ULPI PHY register is written here.

23.9.4.25 USB2_CONTROLLER_1_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 0x174 | Read/Write: R/W | Reset: 0b0000000xx0000000xxx1xx00x0xx010x

Bit	R/W	Reset	Description
31:25	RW	0x0	DA: Device Address. Default = 0000000b. The 7-bit USB device address for the device attached to an immediately downstream of the associated root port. A value of zero indicates no device is present or software support for this feature is not present. This is used by the Controller when sending the LPM token. This field is only valid when the core is operating in host mode. If in device mode it will be read only and always equal to 0000000b.
24:23	RO	X	SSTS: Suspend Status. Default = 00b. These two bits are used by software to determine whether an L1-based suspend request was successful, specifically: 00b - L1 state entered with success. ACK received from peripheral. 01b - NYET received from peripheral. It was not able to enter L1 state this time. 10b - L1 state not supported by peripheral. STALL received. 11b - Peripheral did not respond or an error occurred. The value of this field is only valid when the port resides in the L0 state - that is, the meaning of these bits is invalid whenever bit 7 of this register (SUSP) is one. Ideally, the Controller driver should read this register if it receives an interrupt after issuing a suspend using L1 support. In case of a non-success a port change interrupt will be fired and this field should be checked for a possible L1 failure. This field is only valid when the core is operating in host mode. If in device mode it will be always equal to 00b. 0 = L1STATE_ENTERED 1 = NYET_PERIPH 2 = L1STATE_NOT_SUPPORTED 3 = PERIPH_NORESP_ERR
22	RW	0x0	WKOC: Default = 0b. Wake on Over-current Enable: Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power (PP) is zero. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. 0 = DISABLE 1 = ENABLE
21	RW	0x0	WKDS: Wake on Disconnect Enable: Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power (PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. This bit should not be written to 1 if there is no device connected. After the device disconnect is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
20	RW	0x0	WKCN: Wake on Connect Enable: Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power (PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. This bit should not be written to 1 while the device is connected. After the device connection is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any value other than zero indicates that the port is operating in test mode. Value — Specific Test. 0000b — Not enabled. 0001b — J_ STATE. 0010b — K_STATE. 0011b — SEQ_NAK. 0100b Packet. 0101b — FORCE_ENABLE. 0110b to 1111b — Reserved. Refer to Chapter 7 of the USB Specification Revision 2.0 for details on each test mode.

Bit	R/W	Reset	Description
			0 = NORMAL_OP 1 = TEST_J 2 = TEST_K 3 = TEST_SE0_NAK 4 = TEST_PKT 5 = TEST_FORCE_ENABLE
15:14	RO	X	PIC: Port Indicator Control: This field is not supported in the current implementation. Please use a GPIO if you wish to use Port Indicators.
13	RO	X	PO: Port Owner. Port owner handoff is not implemented in this design, therefore this bit will always be 0.
12	RW	0x1	PP: Port Power: The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0b 0b Read Only. A device controller with no OTG capability does not have port power control switches. 1b 1b/0b RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e., PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). 0 = NOT_POWERED 1 = POWERED
11:10	RO	X	LS: Line state. These bits reflect the current logical levels of the D+ (bit 10) and D- (bit 11) signal lines. The encoding of the bits is: 00b = SE0 01b = J-state 10b = K-state 11b = Undefined The value of this field is undefined if Port Power (PP) is zero in host mode. In host mode, the use of line-state by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of line-state by the device controller driver is not necessary. 0 = SE0 1 = J_STATE 2 = K_STATE 3 = UNDEFINED
9	RW	0x0	SLP: Suspend using L1 - RW. Default = 0b. When this bit is set to '1' and a non-zero Device Address (DA) is specified Controller will instigate L1 entry during suspend (bit 7) and L1 exit during resume (bit 6). When set to zero the Controller will use the legacy (L2) mechanism. Software should only set this bit when the device attached immediately downstream supports L1 transitions. When acting as device, this bit is read-only and set to '1' by the hardware when the Controller enters is L1 state (LPM token received and accepted). Note: HSP is redundant with PSPD(27:26). This bit is not defined in the EHCI specification.
8	RW	0x0	PR: This field is zero if Port Power (PP) is zero. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. When software writes a one to this bit, the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. In Device Mode: This bit is a read-only status bit. Device reset from the USB bus is also indicated in the USBSTS register. 0 = NOT_USB_RESET 1 = USB_RESET

Bit	R/W	Reset	Description
7	RO	X	SUSP: Port suspend. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend When in the suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (i.e., Port enabled bit is a zero), the results are undefined. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: Read Only. This bit is a read-only status bit. 0 = NOT_SUSPEND 1 = SUSPEND
6	RW	0x0	FPR: Force Port Resume. 1= Resume detected/driven on port. 0=No resume (K state) detected/driven on port. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit remains a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time the resume operation to clear the bit when the port control state switches to HS or FS idle. This field is zero if Port Power (PP) is zero in host mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5 ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. Software should ensure that the PHY clock is operational before writing a 1 to this bit to start the resume sequence. This is true for both Device and Host modes. 0 = NO_RESUME 1 = RESUME
5	RO	X	OCC: Over-current Change: Not supported 0 = NO_CHANGE 1 = CHANGE
4	RO	X	OCA: Over-current Active: Not supported 0 = NO_OVER_CURRENT 1 = OVER_CURRENT
3	RW	0x0	PEC: Port Enable/Disable Change: 1=Port enabled/disabled status has changed. 0=No change. In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if Port Power (PP) is zero. In Device mode: The device port is always enabled. (This bit will be zero) 0 = NO_CHANGE 1 = CHANGE
2	RW	0x1	PE: Port Enabled/Disabled: 1=Enable. 0=Disable (default) In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b)

Bit	R/W	Reset	Description
			downstream propagation of data is blocked except for reset. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: The device port is always enabled. (This bit will be one) 0 = PORT_DISABLED 1 = PORT_ENABLED
1	RW	0x0	CSC: Connect Status Change: 1 =Change in Current Connect Status. 0=No change (default) In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power (PP) is zero in host mode. This bit is undefined in device controller mode. 0 = NO_CHANGE 1 = CHANGE
0	RO	X	CCS: Current Connect Status. In Host Mode: 1=Device is present on port. 0=No device is present (default). This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: 1=Attached. 0=Not Attached (default). A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0 = NOT_CONNECTED 1 = CONNECTED

23.9.4.26 USB2_CONTROLLER_1_USB2D_HOSTPC1_DEVLC_0

USB2D Device Mode LPM Behav and Control Register

This register controls the LPM behavior and controls the behavior of each USB port. Only available when in device mode. This register shares the same address with the HOSTPC1 register (which is used only in host mode).

USB2D Host Mode LPM Behav and Control Register

This register controls the LPM behavior and controls the behavior of each USB port. Only available when in host mode. This register shares the same address with the DEVLC register (which is used only in device mode).

Offset: 0x1b4 | Read/Write: R/W | Reset: 0b0000xxx0000000000000xxxxxxxxxxx0

Bit	R/W	Reset	Description
31:29	RW	0x0	PTS: Parallel transceiver select. This bit is not defined in the EHCI specification. 0 = UTMI 1 = RESERVED 2 = ULPI 3 = ICUSB_SER 4 = HSIC
28	RW	0x0	STS: Serial transceiver not selected. This is the only value supported. This bit is not defined in the EHCI specification. 0 = PARALLEL_IF 1 = SERIAL_IF
27	RO	X	PTW: Parallel Transceiver Width. Fixed to 0. This bit is not defined in the EHCI specification. 0 = EIGHT_BIT 1 = RESERVED

Bit	R/W	Reset	Description
26:25	RO	X	PSPD: This register field indicates the speed at which the port is operating. 00 = Full Speed 01 = Low Speed 10 = High Speed. This bit is not defined in the EHCI specification. 0 = FULL_SPEED 1 = LOW_SPEED 2 = HIGH_SPEED 3 = RESERVED
24	RW	0x0	ALPD: Auto Low Power While Disconnect - RW. Default = 0b. If set, this feature will be enabled, and every time the port enters the disconnect state, it will also enter in low power state, disabling the transceiver clock. The behavior will be same as if the PHCD bit was enabled (in fact this bit will be set to 1 as soon as low power mode is enabled).When this field is set the WKCN field of PORTSCx register (Wake on Connect Enable) will also be set. This way the core will wake up in case a connect is detected. There will be a delay between the detection of a disconnect and actually enter in low power mode. This delay can be controlled by writing to the ALPDD field in the register USBMODE 0 = DONT_AUTO_LOW_POWER_WHILE_DISCONNECT 1 = AUTO_LOW_POWER_WHILE_DISCONNECT
23	RW	0x0	PFSC: Port Force Full Speed Connect - RW. Default = 0b. Writing this bit to a '1b' will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with an HS host, hub or device. This bit is not defined in the EHCI specification. 0 = DONT_FORCE_FULL_SPEED 1 = FORCE_FULL_SPEED
22	RW	0x0	PHCD: PHY Low Power Suspend - Clock disable: Writing this bit to a 1 will disable the PHY clock. Writing a 0 enables it. Reading this bit will indicate the status of the PHY clock. NOTE: The PHY clock cannot be disabled if it is being used as the system clock. In device mode, the PHY can be put into Low Power Clock Disable when the device is not running (USBCMD RS=0b) or the host has signaled suspend (PORTSCx SUSP=1b). Low Power Clock Disable will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the Controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low Power Clock Disable is completely under the control of software. 0 = DISABLE 1 = ENABLE
21:20	RW	0x0	LPMX: Auto LPM set - RW. Default = 00b. This bit field is **valid during Host Mode Only**. For Device Mode this is reserved. Value Meaning 00b Disables auto LPM. 01b If there is no activity for a certain number of SOFs the controller will send an LPM token, enter in suspend and issue a port change interrupt. 10b Same as above but without issuing an interrupt. 11b Reserved for futures LPM enhancements. The detection of no activity is based on the absence of response from the downstream device. Because of this limitation this feature should not be used if ISO OUT endpoints are being used (as no handshake is expected).
17	RW	0x0	ASUS: Auto Low Power - RW. Default = 0b. This bit field is **valid during Device Mode Only**. In Host Mode it is part of ELPM field. This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters in suspend state it will also enter in low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	EPLPM: Endpoint for LPM token - RW. Default = 0000b. This bit field is **valid during Host Mode Only**. For Device Mode [19:18] is reserved and 17 is ASUS, while 16 is STL. This sets the endpoint number to which the LPM token will be sent. It will be directly mapped to the ENDP field in the LPM token with the EXT PID.
16	RW	0x0	STL: STALL reply to LPM token - RW. Default = 0b. This bit field is **valid during Device Mode Only**. In Host Mode it is a part of ELPM field. When this bit is set to '1', the Controller will reply always with STALL to all incoming LPM tokens. This bit overrides the LPM NYET bit (NYT). 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
15:12	RW	0x0	LPMFRM: Auto LPM SOF Threshold - RW. Default = 0000b. This bit field is **valid during Host Mode Only**. For Device Mode this is reserved. This holds the SOF counter threshold. When the number of SOF with no activity in between reaches this threshold and the auto LPM is enabled, an LPM token will be sent and the port will enter in suspend state. The SOF counter for this threshold is incremented each 125 µs, even if the port is not in HS operation.
11:1	RO	X	BA: bmAttributtes - RO. Default = 00000000000b. This holds the bmAttributes field of the LPM sub-token received, after the EXT PID token.
0	RW	0x0	NYT_ASUS: NYET reply to LPM token - RW. Default = 0b. This bit is NYT during Device Mode. Details follow: When this bit is '1', the device controller will NYET all the LPM tokens. When this bit is set to '0', the Controller will ACK all the LPM tokens if the STALL bit (STL) is also set to '0'. This bit is ASUS in Host Mode. Details follow: This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters in suspend state it will also enter in low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE

23.9.4.27 USB2_CONTROLLER_1_USB2D_OTGSC_0

USB2D On-The-Go (OTG) Status and Control Register

Offset: 0x1f4 | Read/Write: R/W | Reset: 0bx0000000x0000000xxxxxxxxxx000x00

Bit	R/W	Reset	Description
30	RW	0x0	DPIE: Data Pulse Interrupt Enable. Setting this bit enables the Data pulse interrupt. 0 = DISABLE 1 = ENABLE
29	RW	0x0	ONEMSE: 1 millisecond timer Interrupt enable. Setting this bit enables the 1 millisecond timer interrupt. 0 = DISABLE 1 = ENABLE
28	RW	0x0	BSEIE: B Session End Interrupt Enable. Setting this bit enables the B session end interrupt 0 = DISABLE 1 = ENABLE
27	RW	0x0	BSVIE: B Session Valid Interrupt Enable. Setting this bit enables the B session valid interrupt 0 = DISABLE 1 = ENABLE
26	RW	0x0	ASVIE: A Session Valid Interrupt Enable. Setting this bit enables the A session valid interrupt 0 = DISABLE 1 = ENABLE
25	RW	0x0	AVVIE: A VBus Valid Interrupt Enable. Setting this bit enables the A VBus valid interrupt 0 = DISABLE 1 = ENABLE
24	RW	0x0	IDIE: USB ID Interrupt Enable. Setting this bit enables the USB ID interrupt 0 = DISABLE 1 = ENABLE
22	RW	0x0	DPIS: Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0).PortPower = Off (0). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
21	RW	0x0	ONEMSS: 1 millisecond timer Interrupt Status: This bit is set once every millisecond. Software writes a 1 to clear it. 0 = INT_CLEAR 1 = INT_SET

Bit	R/W	Reset	Description
20	RW	0x0	BSEIS: B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
19	RW	0x0	BSVIS: B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
18	RW	0x0	ASVIS: A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software writes a one to clear this bit. 0 = INT_CLEAR 1 = INT_SET
17	RW	0x0	AVVIS: A VBus Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
16	RW	0x0	IDIS: USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
14	RO	X	DPS: Data Bus Pulsing Status. A 1 indicates data bus pulsing is being detected on the port. 0 = STS_CLEAR 1 = STS_SET
13	RO	X	ONEMST: 1 millisecond timer toggle. This bit toggles once per millisecond 0 = STS_CLEAR 1 = STS_SET
12	RO	X	BSE: B session End. Indicates VBus is below the B session end threshold 0 = STS_CLEAR 1 = STS_SET
11	RO	X	BSV: B Session Valid. Indicates VBus is above the B session valid threshold 0 = STS_CLEAR 1 = STS_SET
10	RO	X	ASV: A Session Valid. Indicates VBus is above the A session valid threshold 0 = STS_CLEAR 1 = STS_SET
9	RO	X	AVV: A VBus Valid. Indicates VBus is above the A VBus valid threshold 0 = STS_CLEAR 1 = STS_SET
8	RO	X	ID: USB ID: 0 = A-device 1 = B-device 0 = A_DEV 1 = B_DEV
5	RW	0x0	IDPU: USB ID Pullup 0 = CLEAR 1 = SET
4	RW	0x0	DP: Data Pulsing. Setting this bit causes the pull-up on DP to be asserted for data pulsing during SRP. 0 = NO_DATA_PULSE 1 = DATA_PULSE
3	RW	0x0	OT: OTG Termination. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM. 0 = NO_OTG_TERM 1 = OTG_TERM
1	RW	0x0	VC: VBUS Charge. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP. 0 = NO_VBUS_CHRG 1 = VBUS_CHRG

Bit	R/W	Reset	Description
0	RW	0x0	VD: VBUS_Discharge. Read/write. Setting this bit causes Vbus to discharge through a resistor. 0 = NO_VBUS_DISCHRG 1 = VBUS_DISCHRG

23.9.4.28 USB2_CONTROLLER_1_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 0x1f8 | Read/Write: R/W | Reset: 0b00000000000000000xxxxxxxxx0xxxxx

Bit	R/W	Reset	Description
31:16	RW	0x0	ALPDD: Auto Low Power While Disconnect Delay. Used when the ALPD field of the HOSTPCx register is set. Defines the delay between disconnect detection and entering in low power mode. The delay is <this register value>*64*100/3 in milliseconds. The maximum value is 65535*64*100/3 = 139,808 ms. The minimum value is 0. Only used in Host mode.
15	RW	0x0	SRT: Shorten USB Reset Time. Software should never set this to 1.
5	RW	0x0	VBPS: VBUS Power Select This can be used by logic that selects between an on-chip Vbus power source (charge pump) and an off-chip source in systems when both are available. Only to be used in Host Mode. Software should not use this.
4	RO	X	SDIS: Stream disable: 1 Streaming is disabled - helpful to avoid overruns/underruns when system load is too high. 0 = STREAM_ENABLE 1 = STREAM_DISABLE
3	RO	X	SLOM: Setup Lockout Mode: In device mode, this bit controls the behavior of the setup lockout mechanism. 0 - Setup lockout is ON (default) 1 Setup lockout is OFF. Firmware requires the use of setup tripwire semaphore in USB2D_USBCMD register. 0 = LOCKOUT_ON 1 = LOCKOUT_OFF
2	RO	X	ES: Endian Select: Note: For this implementation, this should be always set to 0 (little endian). 0 = LITTLE_ENDIAN 1 = RESERVED
1:0	RO	X	CM: Controller Mode: The controller mode will default to an idle state and will need to be initialized to the desired operating mode after reset. This register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register. 00 = Idle [Default]. 01 = Reserved. 10 = Device Controller. 11 = Host Controller. 0 = IDLE 1 = RESERVED 2 = DEVICE_MODE 3 = HOST_MODE

23.9.4.29 USB2_CONTROLLER_1_USB2D_ENDPTNAK_0

USB2D Endpoint NAK register

Offset: 0x200 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTN: TX Endpoint NAK R/WC. Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. 0 = CLEAR 1 = SET
15:0	0x0	EPRN: RX Endpoint NAK R/WC. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. 0 = CLEAR 1 = SET

23.9.4.30 USB2_CONTROLLER_1_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable Register

Offset: 0x204 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTNE: TX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE
15:0	0x0	EPRNE: RX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE

23.9.4.31 USB2_CONTROLLER_1_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 0x208 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15	0x0	ENDPTSETUPSTAT15: Endpoint 15 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
14	0x0	ENDPTSETUPSTAT14: Endpoint 14 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
13	0x0	ENDPTSETUPSTAT13: Endpoint 13 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
12	0x0	ENDPTSETUPSTAT12: Endpoint 12 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
11	0x0	ENDPTSETUPSTAT11: Endpoint 11 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
10	0x0	ENDPTSETUPSTAT10: Endpoint 10 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total

Bit	Reset	Description
		response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
9	0x0	ENDPTSETUPSTAT9: Endpoint 9 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
8	0x0	ENDPTSETUPSTAT8: Endpoint 8 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
7	0x0	ENDPTSETUPSTAT7: Endpoint 7 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
6	0x0	ENDPTSETUPSTAT6: Endpoint 6 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
5	0x0	ENDPTSETUPSTAT5: Endpoint 5 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
4	0x0	ENDPTSETUPSTAT4: Endpoint 4 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
3	0x0	ENDPTSETUPSTAT3: Endpoint 3 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
2	0x0	ENDPTSETUPSTAT2: Endpoint 2 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
1	0x0	ENDPTSETUPSTAT1: Endpoint 1 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
0	0x0	ENDPTSETUPSTAT0: Endpoint 0 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

23.9.4.32 USB2_CONTROLLER_1_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 0x20c | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	PETB15: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
30	0x0	PETB14: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
29	0x0	PETB13: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
28	0x0	PETB12: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
27	0x0	PETB11: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
26	0x0	PETB10: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
25	0x0	PETB9: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
24	0x0	PETB8: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
23	0x0	PETB7: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
22	0x0	PETB6: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
21	0x0	PETB5: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
20	0x0	PETB4: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
19	0x0	PETB3: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
18	0x0	PETB2: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
17	0x0	PETB1: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
16	0x0	PETB0: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
15	0x0	PERB15: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
14	0x0	PERB14: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
13	0x0	PERB13: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
12	0x0	PERB12: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
11	0x0	PERB11: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
10	0x0	PERB10: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
9	0x0	PERB9: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
8	0x0	PERB8: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
7	0x0	PERB7: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
6	0x0	PERB6: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
5	0x0	PERB5: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
4	0x0	PERB4: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
3	0x0	PERB3: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
2	0x0	PERB2: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
1	0x0	PERB1: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
0	0x0	PERB0: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

23.9.4.33 USB2_CONTROLLER_1_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 0x210 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	FETB15: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
30	0x0	FETB14: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
29	0x0	FETB13: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
28	0x0	FETB12: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
27	0x0	FETB11: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
26	0x0	FETB10: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
25	0x0	FETB9: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
24	0x0	FETB8: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
23	0x0	FETB7: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
22	0x0	FETB6: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
21	0x0	FETB5: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
20	0x0	FETB4: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
19	0x0	FETB3: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
18	0x0	FETB2: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
17	0x0	FETB1: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
16	0x0	FETB0: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
15	0x0	FERB15: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
14	0x0	FERB14: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
13	0x0	FERB13: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
12	0x0	FERB12: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
11	0x0	FERB11: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
10	0x0	FERB10: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
9	0x0	FERB9: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
8	0x0	FERB8: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
7	0x0	FERB7: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
6	0x0	FERB6: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
5	0x0	FERB5: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
4	0x0	FERB4: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
3	0x0	FERB3: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
2	0x0	FERB2: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
1	0x0	FERB1: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
0	0x0	FERB0: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

23.9.4.34 USB2_CONTROLLER_1_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 0x214 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	ETBR15: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
30	X	ETBR14: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
29	X	ETBR13: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
28	X	ETBR12: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
27	X	ETBR11: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
26	X	ETBR10: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
25	X	ETBR9: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
24	X	ETBR8: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
23	X	ETBR7: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
22	X	ETBR6: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
21	X	ETBR5: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
20	X	ETBR4: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
19	X	ETBR3: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
18	X	ETBR2: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
17	X	ETBR1: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
16	X	ETBR0: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
15	X	ERBR15: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
14	X	ERBR14: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
13	X	ERBR13: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
12	X	ERBR12: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
11	X	ERBR11: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
10	X	ERBR10: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
9	X	ERBR9: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
8	X	ERBR8: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
7	X	ERBR7: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
6	X	ERBR6: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
5	X	ERBR5: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
4	X	ERBR4: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
3	X	ERBR3: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
2	X	ERBR2: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
1	X	ERBR1: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
0	X	ERBR0: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

23.9.4.35 USB2_CONTROLLER_1_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 0x218 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ETCE15: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
30	0x0	ETCE14: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
29	0x0	ETCE13: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
28	0x0	ETCE12: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
27	0x0	ETCE11: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
26	0x0	ETCE10: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
25	0x0	ETCE9: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
24	0x0	ETCE8: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
23	0x0	ETCE7: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
22	0x0	ETCE6: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
21	0x0	ETCE5: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
20	0x0	ETCE4: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
19	0x0	ETCE3: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
18	0x0	ETCE2: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
17	0x0	ETCE1: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
16	0x0	ETCE0: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
15	0x0	ERCE15: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
14	0x0	ERCE14: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
13	0x0	ERCE13: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
12	0x0	ERCE12: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
11	0x0	ERCE11: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
10	0x0	ERCE10: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
9	0x0	ERCE9: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
8	0x0	ERCE8: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
7	0x0	ERCE7: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
6	0x0	ERCE6: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
5	0x0	ERCE5: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
4	0x0	ERCE4: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
3	0x0	ERCE3: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
2	0x0	ERCE2: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
1	0x0	ERCE1: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
0	0x0	ERCE0: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

23.9.4.36 USB2_CONTROLLER_1_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 0x21c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23	X	TXE: TX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19:18	X	TXT: TX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
16	X	TXS: TX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL
7	X	RXE: RX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
3:2	X	RXT: RX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
0	X	RXS: RX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL

23.9.4.37 USB2_CONTROLLER_1_USB2D_ENDPTCTRL1_0

USB2D Endpoint Control 1 Register

Offset: 0x220 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

Bit	R/W	Reset	Description
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above, 0 = EP_OK 1 = EP_STALL

23.9.4.38 USB2_CONTROLLER_1_USB2D_ENDPTCTRL2_0

USB2D Endpoint Control 2 Register

Offset: 0x224 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.

Bit	R/W	Reset	Description
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.39 USB2_CONTROLLER_1_USB2D_ENDPTCTRL3_0

USB2D Endpoint Control 3 Register

Offset: 0x228 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR

Bit	R/W	Reset	Description
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.40 USB2_CONTROLLER_1_USB2D_ENDPTCTRL4_0

USB2D Endpoint Control 4 Register

Offset: 0x22c | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ

Bit	R/W	Reset	Description
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.41 USB2_CONTROLLER_1_USB2D_ENDPTCTRL5_0

USB2D Endpoint Control 5 Register

Offset: 0x230 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ

Bit	R/W	Reset	Description
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.42 USB2_CONTROLLER_1_USB2D_ENDPTCTRL6_0

USB2D Endpoint Control 6 Register

Offset: 0x234 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.43 USB2_CONTROLLER_1_USB2D_ENDPTCTRL7_0

USB2D Endpoint Control 7 Register

Offset: 0x238 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.44 USB2_CONTROLLER_1_USB2D_ENDPTCTRL8_0

USB2D Endpoint Control 8 Register

Offset: 0x23c | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.45 USB2_CONTROLLER_1_USB2D_ENDPTCTRL9_0

USB2D Endpoint Control 9 Register

Offset: 0x240 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.46 USB2_CONTROLLER_1_USB2D_ENDPTCTRL10_0

USB2D Endpoint Control 10 Register

Offset: 0x244 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.47 USB2_CONTROLLER_1_USB2D_ENDPTCTRL11_0

USB2D Endpoint Control 11 Register

Offset: 0x248 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.48 USB2_CONTROLLER_1_USB2D_ENDPTCTRL12_0

USB2D Endpoint Control 12 Register

Offset: 0x24c | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.49 USB2_CONTROLLER_1_USB2D_ENDPTCTRL13_0

USB2D Endpoint Control 13 Register

Offset: 0x250 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.50 USB2_CONTROLLER_1_USB2D_ENDPTCTRL14_0

USB2D Endpoint Control 14 Register

Offset: 0x254 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.4.51 USB2_CONTROLLER_1_USB2D_ENDPTCTRL15_0

USB2D Endpoint Control 15 Register

Offset: 0x258 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.5 USB2 Controller Interface Registers

These are used to generate the actual RTL registers for USB2 controller interface.

ULPIS2S resets to be used as follows:

- Assert ULPIS2S_SLV0_CLAMP_XMIT to ensure that the signals between SLV0 and the line simulator are clean.
- Assert ULPIS2S_SLV0_RESET
- Do a functional asynchronous reset of the ChipIdea controller
- Deassert ULPIS2S_SLV0_RESET
- Deassert ULPIS2S_SLV0_CLAMP_XMIT

NULPI_SLV1_RESET should be used in the same way if there is some explicit way to reset the external ULPI controller (which is not necessarily the case). There is usually no reason to assert ULPIS2S_LINE_RESET.

23.9.5.1 USB2_IF_USB_SUSP_CTRL_0

This register controls the suspend and resume behavior of the USB controller/PHY.

USB Suspend Control Register

Offset: 0x400 | Read/Write: R/W | Reset: 0bxxxxx00111110000x1001000xx000000

Bit	R/W	Reset	Description
26	RW	0x0	FAST_WAKEUP_RESP: Enable Fast Response from UTMIP PHY for a Remote Wakeup request from device. This is used only for cases where wakeup response needs to be within 1ms of spec. Used for Host mode ONLY. 0 = DISABLE 1 = ENABLE
25	RW	0x0	UTMIP_SUSPL1_SET: Enable SuspendL1 for UTMIP PHY Enabling this will only cutoff clocks to the UTMIP logic. The USB PLLs, PllU and UTMIP PLL will still be running. 0 = DISABLE 1 = ENABLE
24	RW	0x1	ULPI_PADS_CLKEN_RESET: Async reset for the synchronizers that are used in the external and loopback ULPI 60 MHz clock. 0 = DISABLE 1 = ENABLE
23	RW	0x1	ULPI_PADS_RESET: Async reset for trimmers and line state logic that is implemented in the pad macros. 0 = DISABLE 1 = ENABLE
22	RW	0x1	ULPIS2S_LINE_RESET: Async reset of the line simulator logic that sits between the two virtual PHYs (active high). The ULPI PHY registers should be programmed while ULPI is in reset. 0 = DISABLE 1 = ENABLE
21	RW	0x1	ULPIS2S_SLV1_RESET: Async reset of the SLV1 ULPI logic. This corresponds to resetting the virtual PHY that is connected to the internal ULPI controller (active high). The ULPI PHY registers should be programmed while ULPI is in reset. 0 = DISABLE 1 = ENABLE
20	RW	0x1	ULPIS2S_SLV0_RESET: Async reset of the SLV0 ULPI logic. This corresponds to resetting the virtual PHY that is connected to the external ULPI controller. (active high). The ULPI PHY registers should be programmed while ULPI is in reset. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
19	RW	0x0	UHSIC_PHY_ENB: Enable UHSIC PHY mode. Set this to 1 if using UHSIC PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
18:16	RW	0x0	USB_WAKEUP_DEBOUNCE_COUNT: USB PHY wakeup debounce counter USB will debounce any wakeup event by the number of clocks programmed in this counter. A value of 0 results in no debounce.
14	RW	0x1	UHSIC_RESET: Reset going to UHSIC PHY (active high). This should be set to 1 whenever programming the UHSIC config registers. It should be cleared to 0 after the programming of UHSIC config registers is done. UHSIC config registers should be programmed only once before doing any transactions on UHSIC. The UHSIC PHY registers should be programmed while UHSIC is in reset. 0 = DISABLE 1 = ENABLE
13	RW	0x0	ULPI_PHY_ENB: Enable ULPI PHY mode. Set this to 1 if using null or link ULPI PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
12	RW	0x0	UTMIP_PHY_ENB: Enable UTMIP PHY mode Set this to 1 if using UTMIP PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
11	RW	0x1	UTMIP_RESET: Reset going to UTMIP PHY (active high). This should be set to 1 whenever programming the UTMIP config registers. It should be cleared to 0 after the programming of UTMIP config registers is done. UTMIP config registers should be programmed only once before doing any transactions on UTMIP. The UTMIP PHY registers should be programmed while UTMIP is in reset. 0 = DISABLE 1 = ENABLE
10	RW	0x0	USB_SUSP_POL: Polarity of the suspend signal going to UHSIC PHY. 0 = Active low (default) 1 = Active high This should not be changed by software. 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
9	RW	0x0	USB_PHY_CLK_VALID_INT_ENB: USB PHY clock valid interrupt enable. If this bit is enabled, interrupt is generated whenever USB clocks are resumed from a suspend. 0 = DISABLE 1 = ENABLE
8	RO	0x0	USB_PHY_CLK_VALID_INT_STS: USB PHY clock valid interrupt status. This bit is set whenever USB PHY clock is woken up from suspend. Software must write a 1 to clear this bit. 0 = UNSET 1 = SET
7	RO	X	USB_PHY_CLK_VALID: USB PHY clock valid status. This bit indicates whether the USB PHY is generating a valid clock to the USB controller. If USB PHY clock is running, this bit is set to 1, else it is set to 0. 0 = UNSET 1 = SET
6	RO	X	USB_CLKEN: USB AHB clock enable status. Indicates whether the AHB clock to the USB controller is enabled or not. If AHB clock to USB controller is enabled, this bit is set to 1, else it is set to 0. NOTE: even when this is set to 0, all essential blocks that are required to resume USB clocks from suspend will be active and their AHB clock will not be suspended. 0 = UNSET 1 = SET
5	RW	0x0	USB_SUSP_CLR: Suspend Clear Software must write a 1 to this bit to bring the PHY out of suspend mode. This is used when the software stops the PHY clock during suspend and then wants to initiate a resume. Software should also write 0 to clear it. NOTE: It is required that software generate a positive pulse on this bit to guarantee proper operation. 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
4	RW	0x0	USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect Enable (device mode). When enabled (1), USB device will wake up from suspend on a disconnect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
3	RW	0x0	USB_WAKE_ON_CNNT_EN_DEV: Wake on Connect Enable (device mode). When enabled (1), USB device will wake up from suspend on a connect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable. If this bit is enabled, the USB will wake up from suspend whenever a resume event is detected on USB. This is valid for both USB device and USB host modes. 0 = DISABLE 1 = ENABLE
1	RW	0x0	USB_WAKEUP_INT_ENB: USB wakeup interrupt enable. If this bit is enabled, interrupt is generated whenever USB wakeup event is generated. 0 = DISABLE 1 = ENABLE
0	RO	0x0	USB_WAKEUP_INT_STS: USB wakeup interrupt status. This bit is set whenever the USB wakes up from suspend (a wakeup event is generated). Software must write a 1 to clear this bit. Note that during the wakeup sequence, PHY clocks will be resumed from suspend. Software can check when the PHY clocks are resumed by reading the bit USB_PHY_CLK_VALID. There is also a separate interrupt generated when PHY clock is resumed if USB_PHY_CLK_VALID_INT_EN is set. During the wakeup sequence, first USB_WAKEUP_INT_STS will be set, and it will take some time for the PHY clock to resume, which can be detected by checking USB_PHY_CLK_VALID. 0 = UNSET 1 = SET

23.9.5.2 USB2_IF_USB_ULPIS2S_CTRL_0

This register is used to set up parameters for ULPI null PHY mode.

Note: Current ULPI configuration registers use settings that support NV software drivers and should NOT be modified. If designing custom drivers, consult your NV representative prior to modifying these registers

ULPI NULL PHY Control Register

Offset: 0x418 | Read/Write: R/W | Reset: 0bxxxxxxxx0000xx0000000000xxxx0000

Bit	Reset	Description
23:20	0x0	ULPIS2S_CLAMP_LINE_DRIVE: The line drive value that should be sent into the line simulator during transmit clamping. The suggested value is 0x1: tri-state.
17	0x0	ULPIS2S_SLV1_CLAMP_XMIT: When set to 1, the outputs of the SLV1 transmit state machine are clamped to 0. This bit should be set before the SLV0 async reset is asserted to prevent meta-stability issues in the line simulator.
16	0x0	ULPIS2S_SLV0_CLAMP_XMIT: When set to 1, the outputs of the SLV0 transmit state machine are clamped to 0. This bit should be set before the SLV0 async reset is asserted to prevent meta-stability issues in the line simulator.
15	0x0	ULPIS2S_DISABLE_STP_PU: When set to 1 and in ULPIS2S mode, the pullup on the STP pin will NOT be active, even if the remote LINK asks to do so. In this case, an external pullup resistor would be required to ensure valid levels when the remote link is not powered.
14	0x0	ULPIS2S_SUPPORT_HS_KEEP_ALIVE: When enabled, the PHY will support HS KeepAlive packets. In that case, this would be the only thing that is supported in Opmode3. All other Opmode3 generate packets are not supported under any circumstances. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
13	0x0	ULPIS2S_DISCON_DONT_CHECK_SE0: When enabled, the disconnect detection logic will only check that that the other side is 'driving' tri-state. It will not check whether or not the local side is driving SE0. 0 = DISABLE 1 = ENABLE
12	0x0	ULPIS2S_FORCE_ULPI_CLK_OUT: When enabled and ULPIS2S_ENA is ENABLED, the external ULPI_CLOCK pad will always carry the internal 60MHz clock, even if the interface is in shutdown mode. 0 = DISABLE 1 = ENABLE
11:8	0x0	ULPIS2S_SPARE: Reserved bits.
3	0x0	ULPIS2S_PLLU_MASTER_BLASTER60: When enabled, the PLLU 60MHz clock will be forced on. 0 = DISABLE 1 = ENABLE
2	0x0	ULPIS2S_SUPPORT_DISCONNECT: When disabled, the PHY will never detect a Disconnect. 0 = DISABLE 1 = ENABLE
1	0x0	ULPIS2S_SLV1_FORCE_DEVICE: When disabled, the slave port that is connected to the pins can be programmed to be host or a device depending on the value of the DpPulldown and DmPulldown bits in the OTG_CTRL ULPI register. When enabled, the values of those bits in the OTG_CTRL register is ignored and the port will always behave like a device. 0 = DISABLE 1 = ENABLE
0	0x0	ULPIS2S_ENA: When enabled, the ULPI link interface coming out of the USB2 controller enters a NULL PHY with two slaves. As a result the external pins will have a slave ULPI interface. When disabled, the ULPI link interface coming out of the USB2 controller go straight to the pins. 0 = DISABLE 1 = ENABLE

23.9.5.3 USB2_IF_USB_ULPIS2S_SLV1_ID_0

This register controls the product and vendor ID fields for ULPI null PHY presented to external ULPI master.

Note: Current ULPI configuration registers use settings that support NV software drivers and should NOT be modified. If designing custom drivers, consult your NV representative prior to modifying these registers

Offset: 0x41c | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	ULPIS2S_SLV1_VENDOR_ID: PHY vendor_id as seen by the external ULPI master
15:0	0x0	ULPIS2S_SLV1_PRODUCT_ID: PHY product_id as seen by the external ULPI master

23.9.5.4 USB2_IF_USB_INTER_PKT_DELAY_CTRL_0

Inter Packet Delay Control

This register controls the interpacket delay between two consecutive transmit packets in HS mode. This only applies to host mode as device never transmits two packets in a row.

Offset: 0x420 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0010010

Bit	Reset	Description
6:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. This is valid only for UHSIC PHY. Software should not change this.

23.9.5.5 USB2_IF_USB_RSM_DLY_0

USB Controller Resume Signalling Delay

Offset: 0x490 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0110100101111000

Bit	Reset	Description
15:0	0x6978	TIME_TO_RESUME: Send the resume back in number of 60 MHz cycles. Default gives 900 µs delay. Only applicable in host mode.

23.9.5.6 USB2_IF_SPARE_0

For ICUSB PADCTLS

Spare Register

Offset: 0x498 | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:16	0xffff	SPARE_HI: Spare register bits, defaulted to 1. SPARE_HI[0] - usb_port_reset_fix_en. Enable port-reset fix SPARE_HI[1] - hsic_conn_det_feature_enable. enable HSIC connect detection
15:0	0x0	SPARE_LO: Spare register bits, defaulted to 0. SPARE_LO[4] - usb_hs_rsm_eop_fix_en. Enable HS_RESUME EOP fix SPARE_LO[5] - usb_port_suspend_fix_en

23.9.5.7 USB2_IF_ULPI_DIR_OVERRIDE_0

ULPI Override

Offset: 0x49c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1

Bit	Reset	Description
0	0x1	ULPI_DIR_OVERRIDE: By default, this bit is set. This will override ulpi_dir; i.e., when this bit is set, ulpi_dir is always asserted. Software needs to explicitly clear this bit, once slv1 Lp0 context is restored.

23.9.5.8 USB2_IF_USB2_NEW_CONTROL_0

USB Coherency and Memory Alignment Controls

Offset: 0x4c0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx00001111xxxxxx00

Bit	Reset	Description
15:8	0xf	REQUEST_EXPIRY_COUNTER: Time to wait for coalescing the request.
1	0x0	MEM_ALIGNMENT_MUX_EN: Mux to select between Tegra 3 and Tegra 4 code 0 = DISABLE 1 = ENABLE
0	0x0	COHERENCY_EN: Enable fence mechanism. 0 = DISABLE 1 = ENABLE

23.9.6 UHSIC Configuration Registers

Note: Current HSIC configuration registers use settings that support NV software drivers and should NOT be modified. If designing custom drivers, consult your NV representative prior to modifying these registers

23.9.6.1 USB2_UHSIC_PLL_CFG0_0

UHSIC PHY PLL Configuration Register 0

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

Offset: 0xc00 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
0	0x0	UHSIC_PLL_SPARE: Reserved

23.9.6.2 USB2_UHSIC_PLL_CFG1_0

UHSIC PLL and PLLU Configuration Register 1

PLL CONFIGURATION and PARAMETERS

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

Crystal clock -> enters PLLU to generate 12 MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock

The following parameters control the bring-up of the PLLs:

Coming out of reset or suspend

PllUOnState: start pllu_enable_count and pll_lock_count

Wait ~1 µs to actually enable the PLLU (pllu_enable_count == ClkXtal * PLLU_ENABLE_DLY_COUNT * 8)

Wait ~1ms until PLLU is actually stable (pll_lock_count == ClkXtal * PLLU_STABLE_COUNT * 256) => USB_PHY PLL_ENABLE

Number for a 19.2MHz crystal clock

- PLLU_ENABLE_DLY_COUNT[4:0] = 1 * 19.2 / 8 = 2.4 = 3 = 0x03
- PLLU_STABLE_COUNT[11:0] = 1000 * 19.2 / 256 = 75 = 0x4b (Note: currently defaults to 0x600)
- XTAL_FREQ_COUNT[11:0] = 2500 * 19.2 / 256 = 187 = 0xbb

This register is used to configure the PHY PLL contained in the UHSIC module as well as the PLLU power up and down.

Offset: 0xc04 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx0001100000011000000

Bit	Reset	Description
18:14	0x3	UHSIC_PLLU_ENABLE_DLY_COUNT: 1 µs / (1/19.2MHz) = 19 / 8 = 2.36 = 3
13	0x0	UHSIC_FORCE_PLLU_POWERUP
12	0x0	UHSIC_FORCE_PLLU_POWERDOWN
11:0	0xc0	UHSIC_XTAL_FREQ_COUNT: 2.5ms / (1/19.2MHz) = 48000 / 256 = 187 = 0xBB

23.9.6.3 USB2_UHSIC_HSRX_CFG0_0

UHSIC High Speed Receive Config 0

Offset: 0xc08 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx0010100111000111000

Bit	Reset	Description
18	0x0	UHSIC_NO_STRIPPING: Do not strip incoming data
17:13	0xa	UHSIC_IDLE_WAIT: Number of idle cycles to declare IDLE.
12:8	0xe	UHSIC_ELASTIC_OVERRUN_LIMIT
7	0x0	UHSIC_ELASTIC_OVERRUN_DISABLE: Do not declare overrun errors until overflow of FIFO
6:2	0xe	UHSIC_ELASTIC_UNDERRUN_LIMIT
1	0x0	UHSIC_ELASTIC_UNDERRUN_DISABLE: Do not declare underrun errors
0	0x0	UHSIC_PASS_FEEDBACK: Pass through the feedback, do not block it.

23.9.6.4 USB2_UHSIC_HSRX_CFG1_0

UHSIC High speed receive config 1

Offset: 0xc0c | Read/Write: R/W | Reset: 0bxxxxxxxx10001000001010010010011

Bit	Reset	Description
23:20	0x8	UHSIC_TX_BLOCK_CNT: Controls how long after the end of transmission the receive path is blocked
19:14	0x20	UHSIC_RX_STROBE_DLY_TRIMMER: Number of delays cells between UH_RX_STROBE and RxStrobeClk in zero cycle path
13:9	0x14	UHSIC_INPUT_FIFO_DEPTH: Depth of the 2-bit wide input FIFO. Maximum depth is 20. Can be tuned
8	0x1	UHSIC_LINE_STATE_RESUME_FAKE_SE0: When enabled, send an SE0 for 2 LS symbols at the end of ResumeK
7	0x0	UHSIC_LINE_STATE_BYPASS: Bypass Line State reclocking logic
6	0x0	UHSIC_EARLY_LINE_STATE_FILTER: Assumes line state filtering table is inclusive, not exclusive
5:1	0x9	UHSIC_HS_SYNC_START_DLY: How long to wait before start of sync launches RxActive
0	0x1	UHSIC_HS_ALLOW_KEEP_ALIVE: Allow Keep Alive packets

23.9.6.5 USB2_UHSIC_TX_CFG0_0

UHSIC Transmit Config Signals

Offset: 0xc10 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx1000000000

Bit	Reset	Description
9	0x1	UHSIC_HS_READY_WAIT_FOR_VALID
8	0x0	UHSIC_PACKET_INVERT_DATA: Invert data during a regular packet
7	0x0	UHSIC_PACKET_FORCE_STROBE_LOW: Force STROBE low during a regular instead of toggling it
6	0x0	UHSIC_HS_POSTAMBLE_OUTPUT_ENABLE: output enable turns off after 1 cycle
5	0x0	UHSIC_SIE_RESUME_ON_LINESTATE: SIE, not macrocell, detects LineState change to resume
4	0x0	UHSIC_SOF_ON_NO_STUFF: SOF when OpMode 3 -- perhaps, when sending controller made packets
3	0x0	UHSIC_SOF_ON_NO_ENCODE: SOF when OpMode 2 -- not likely, for Chirp

Bit	Reset	Description
2	0x0	UHSIC_NO_STUFFING: No bit stuffing, static programming
1	0x0	UHSIC_NO_ENCODING: No encoding, static programming
0	0x0	UHSIC_NO_SYNC_NO_EOP: Do not send SYNC or EOP

23.9.6.6 USB2_UHSIC_MISC_CFG0_0

UHSIC Miscellaneous Configurations

Offset: 0xc14 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxx00100010011101000111 10

Bit	Reset	Description
20	0x0	UHSIC_DISABLE_BUSRESET: When 1, the PHY will not send out BusReset during XcvrSelect0, TermSelect0 and Opmode2. It will send out a non-bit-stuffed, non-encoded packet instead.
19	0x0	UHSIC_FORCE_TERMSEL: Value to be forced on TermSelect when FORCE_XCVR_MODE is set.
18	0x1	UHSIC_EXTEND_BK_ACTIVE: Drive bus keeper one cycle longer when going out of IDLE
17:16	0x0	UHSIC_FORCE_XCVRSEL: Value to be forced on XcvrSelect when FORCE_XCVR_MODE is set.
15	0x0	UHSIC_FORCE_XCVR_MODE: 1: Force the values of XcvrSelect and TermSelect via config bits instead of via the controller
14	0x1	UHSIC_SYMMETRIC_CONNECT_DATA 0: DATA goes high before STROBE goes low and low before STROBE goes high. 1: DATA goes high before STROBE goes low and goes low *after* STROBE goes high.
13	0x0	UHSIC_ASYNC_CONNECT_DATA 0: DATA keeps setup and hold requirements during CONNECT. 1: DATA moves together with STROBE
12	0x0	UHSIC_LONG_CONNECT_STROBE 0: STROBE is 2 periods long during connect. 1: STROBE is 3 periods long during connect
11	0x1	UHSIC_ACTIVE_BK_DRIVE_RX: 1: Use RX state (EOP, etc.) to determine starting time to drive bus keeper instead of waiting for IDLE detection.
10	0x1	UHSIC_ACTIVE_BK_DRIVE_TX: 1: Use TX state to determine starting time to drive bus keeper instead of waiting for IDLE detection.
9	0x1	UHSIC_DETECT_SHORT_IDLE 0: Use 3 edges (negative and positive) to detect an idle state on the line. 1: use 4 edges.
8	0x0	UHSIC_DETECT_SHORT_CONNECT 0: Use 3 edges (negative and positive) to detect a connect state on the line. 1: Use 4 edges.
7	0x1	UHSIC_SUSPEND_EXIT_ON_EDGE: Suspend exit requires edge or simply a value.
6:5	0x0	UHSIC_INJECT_ERROR_TYPE: Force error insertion into RX path. (Used for IOBIST.) 0 = DISABLE 1 = BIT_ERR 2 = RX_ERR 3 = BIT_RX_ERR
4:2	0x3	UHSIC_STABLE_COUNT: Number of crystal clock cycles of signal not changing to consider stable.
1	0x1	UHSIC_STABLE_ALL: Determines if all signals need to be stable to not change a config.
0	0x0	UHSIC_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

23.9.6.7 USB2_UHSIC_MISC_CFG1_0

UHSIC Miscellaneous Configurations

Offset: 0xc18 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxx100001100000000010

Bit	Reset	Description
17	0x1	UHSIC_PHY_XTAL_CLOCKEN: Selects whether or not to enable the crystal clock in the module.
16:15	0x0	UHSIC_OBS_SEL: Selects which one of 4 observation vectors is presented on the observation bus
14	0x0	UHSIC_FORCE_IOBIST_CLK_ON: Always enable IoBist CLK60. This is required when you want to use RX_ERROR_CNT_EN.
13:2	0x600	UHSIC_PLLU_STABLE_COUNT: PLLU frequency lock delay.
1	0x1	UHSIC_RX_ERROR_CNT_CLR: Clear IOBST RxError counter.
0	0x0	UHSIC_RX_ERROR_CNT_EN: Enable IOBIST RxError counter when not in IOBIST mode. Allows one to read out the number of errors via JTAG during normal operation

23.9.6.8 USB2_UHSIC_PADS_CFG0_0

UHSIC Pads settings

Offset: 0xc1c | Read/Write: R/W | Reset: 0b00000000100010001000100010001000

Bit	Reset	Description
31:24	0x0	UHSIC_HSIC_OPT: Spare config bits
23:20	0x8	UHSIC_TX_SLEWN: Output slew rate (fall time) adjustment
19:16	0x8	UHSIC_TX_SLEWP: Output slew rate (rise time) adjustment
15:12	0x8	UHSIC_TX_RTUNEN: Fine tuned 50 ohm termination resistor for NMOS driver
11:8	0x8	UHSIC_TX_RTUNEP: Fine tuned 50 ohm termination resistor for PMOS driver
7:4	0x8	UHSIC_TX_RTERMN: Output impedance adjustment for NMOS driver
3:0	0x8	UHSIC_TX_RTERMP: Output impedance adjustment for PMOS driver

23.9.6.9 USB2_UHSIC_PADS_CFG1_0

UHSIC Pads Settings

Offset: 0xc20 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxx0011001111101

Bit	Reset	Description
12	0x0	UHSIC_RPU_STROBE: Enable pull up on IO_STROBE
11	0x0	UHSIC_RPU_DATA: Enable pull up on IO_DATA
10	0x1	UHSIC_RPD_STROBE: Enable pull down on IO_STROBE
9	0x1	UHSIC_RPD_DATA: Enable pull down on IO_DATA
8	0x0	UHSIC_LPBK: Internal digital loopback
7	0x0	UHSIC_RX_SEL 0: Differential read buffers 1: Single-ended buffers
6	0x1	UHSIC_PD_ZI: Power-down single ended receiver
5	0x1	UHSIC_PD_RX: Power-down receiver

Bit	Reset	Description
4	0x1	UHSIC_PD_TRK: Power down tracking circuit
3	0x1	UHSIC_PD_TX: Power-down transmitter
2	0x1	UHSIC_PD_BG: Power-down band-gap and bias generator
1	0x0	UHSIC_IDDQ: Shut down analog blocks for IDDQ testing
0	0x1	UHSIC_AUTO_RTERM_EN: Enable auto-termination

23.9.6.10 USB2_UHSIC_CMD_CFG0_0

Determine startup behavior.

When AUTO_NEGOTIATE == 0

- Firmware is supposed to take care of things.

HOST

if Opmode1: do not drive anything

else

Initial power up:

- Asynchronously drive 00
- After a few crystal clocks, enable bus keepers

Offset: 0xc24 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000101

Bit	Reset	Description
6	0x0	UHSIC_FORCE_ACTIVATED: Force PHY into activated state without connect handshake (both host and device)
5	0x0	UHSIC_PRETEND_CONNECT_DETECT: While in HOST mode, act as if the input stage has seen a CONNECT pulse from the external PHY
4	0x0	UHSIC_FORCE_RESET: While in HOST mode, force global state machine into RESET state
3	0x0	UHSIC_FORCE_CONNECT: Upon rising value of this bit, force device to send connect. Only useful when AUTO_CONNECT is disabled.
2	0x1	UHSIC_AUTO_CONNECT: As device, automatically send Connect during activation.
1	0x0	UHSIC_FORCE_ACTIVATE: Upon rising value of this bit, instruct state machine to go into activation mode. Only useful when AUTO_ACTIVATE is disabled.
0	0x1	UHSIC_AUTO_ACTIVATE: Upon power up, automatically move to activation mode and start going through the connect procedure.

23.9.6.11 USB2_UHSIC_STAT_CFG0_0

Offset: 0xc28 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
31:16	RO	X	UHSIC_CALIOUT
15:8	RO	X	UHSIC_SPARE_STATUS
2:1	RO	X	UHSIC_BUS_STATE
0	RW	0x0	UHSIC_CONNECT_DETECT

23.9.6.12 USB2_UHSIC_SPARE_CFG0_0

UTMIP Spare Configurations and Spare Configuration Bits

Offset: 0xc2c | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:0	-65536	UHSIC_SPARE: Bit 0 : HS_RX_IPG_ERROR_ENABLE Bit 1 : HS_RX_FLUSH_ALAP Bit 2 : FORCE_TRIM_ZERO Bits 7 :4 : RX_DATA_TRIM[3:0] Bits 11:8 : RX_STROBE_TRIM[3:0] Bit 12 : FORCE_BK_ON Bit 13 : BYPASS_INIT_BLOCK Bit 14 : FORCE_SM_IDLE Bit 15 : FORCE_BK_OFF

23.9.6.13 USB2_UHSIC_MISC_STS0_0

UHSIC SPARE Fuse Value

Offset: 0xc30 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
0	X	UHSIC_TX_HS_VLD: Indicates when the controller is driving on the bus

23.9.6.14 USB2_UHSIC_PMC_WAKEUP0_0

UHSIC PMC Wakeup Value

Offset: 0xc34 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
1	RO	X	UHSIC_LINE_WAKEUP_EVENT_INT_STS: Indicates the status of the PMC wakeup event
0	RW	0x0	UHSIC_LINE_WAKEUP_EVENT_INT_ENB: Interrupt mask for the PMC wakeup event

23.9.6.15 USB2_QH_USB2D_QH_EP_0_OUT_0

USB2D Queue Head for OUT Endpoint 0

Offset: 0x1000 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 0. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 0.

23.9.6.16 USB2_QH_USB2D_QH_EP_0_IN_0

USB2D Queue Head for IN Endpoint 0

Offset: 0x1040 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 0. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 0.

23.9.6.17 USB2_QH_USB2D_QH_EP_1_OUT_0

USB2D Queue Head for OUT Endpoint 1

Offset: 0x1080 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 1. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 1.

23.9.6.18 USB2_QH_USB2D_QH_EP_1_IN_0

USB2D Queue Head for IN Endpoint 1

Offset: 0x10c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 1. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 1.

23.9.6.19 USB2_QH_USB2D_QH_EP_2_OUT_0

USB2D Queue Head for OUT Endpoint 2

Offset: 0x1100 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 2. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 2.

23.9.6.20 USB2_QH_USB2D_QH_EP_2_IN_0

USB2D Queue Head for IN Endpoint 2

Offset: 0x1140 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 2. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 2.

23.9.6.21 USB2_QH_USB2D_QH_EP_3_OUT_0

USB2D Queue Head for OUT Endpoint 3

Offset: 0x1180 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 3. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 3.

23.9.6.22 USB2_QH_USB2D_QH_EP_3_IN_0

USB2D Queue Head for IN Endpoint 3

Offset: 0x11c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 3. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 3.

23.9.6.23 USB2_QH_USB2D_QH_EP_4_OUT_0

USB2D Queue Head for OUT Endpoint 4

Offset: 0x1200 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 4. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 4.

23.9.6.24 USB2_QH_USB2D_QH_EP_4_IN_0

USB2D Queue Head for IN Endpoint 4

Offset: 0x1240 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 4. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 4.

23.9.6.25 USB2_QH_USB2D_QH_EP_5_OUT_0

USB2D Queue Head for OUT Endpoint 5

Offset: 0x1280 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 5. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 5.

23.9.6.26 USB2_QH_USB2D_QH_EP_5_IN_0

USB2D Queue Head for IN Endpoint 5

Offset: 0x12c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 5. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 5.

23.9.6.27 USB2_QH_USB2D_QH_EP_6_OUT_0

USB2D Queue Head for OUT Endpoint 6

Offset: 0x1300 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 6. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 6.

23.9.6.28 USB2_QH_USB2D_QH_EP_6_IN_0

USB2D Queue Head for IN Endpoint 6

Offset: 0x1340 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 6. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 6.

23.9.6.29 USB2_QH_USB2D_QH_EP_7_OUT_0

USB2D Queue Head for OUT Endpoint 7

Offset: 0x1380 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 7. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 7.

23.9.6.30 USB2_QH_USB2D_QH_EP_7_IN_0

USB2D Queue Head for IN Endpoint 7

Offset: 0x13c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 7. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 7.

23.9.6.31 USB2_QH_USB2D_QH_EP_8_OUT_0

USB2D Queue Head for OUT Endpoint 8

Offset: 0x1400 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 8. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 8.

23.9.6.32 USB2_QH_USB2D_QH_EP_8_IN_0

USB2D Queue Head for IN Endpoint 8

Offset: 0x1440 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 8. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 8.

23.9.6.33 USB2_QH_USB2D_QH_EP_9_OUT_0

USB2D Queue Head for OUT Endpoint 9

Offset: 0x1480 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 9. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 9.

23.9.6.34 USB2_QH_USB2D_QH_EP_9_IN_0

USB2D Queue Head for IN Endpoint 9

Offset: 0x14c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 9. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 9.

23.9.6.35 USB2_QH_USB2D_QH_EP_10_OUT_0

USB2D Queue Head for OUT Endpoint 10

Offset: 0x1500 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 10. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 10.

23.9.6.36 USB2_QH_USB2D_QH_EP_10_IN_0

USB2D Queue Head for IN Endpoint 10

Offset: 0x1540 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 10. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 10.

23.9.6.37 USB2_QH_USB2D_QH_EP_11_OUT_0

USB2D Queue Head for OUT Endpoint 11

Offset: 0x1580 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 11. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 11.

23.9.6.38 USB2_QH_USB2D_QH_EP_11_IN_0

USB2D Queue Head for IN Endpoint 11

Offset: 0x15c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 11. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 11.

23.9.6.39 USB2_QH_USB2D_QH_EP_12_OUT_0

USB2D Queue Head for OUT Endpoint 12

Offset: 0x1600 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 12. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 12.

23.9.6.40 USB2_QH_USB2D_QH_EP_12_IN_0

USB2D Queue Head for IN Endpoint 12

Offset: 0x1640 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 12. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 12.

23.9.6.41 USB2_QH_USB2D_QH_EP_13_OUT_0

USB2D Queue Head for OUT Endpoint 13

Offset: 0x1680 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 13. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 13.

23.9.6.42 USB2_QH_USB2D_QH_EP_13_IN_0

USB2D Queue Head for IN Endpoint 13

Offset: 0x16c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 13. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 13.

23.9.6.43 USB2_QH_USB2D_QH_EP_14_OUT_0

USB2D Queue Head for OUT Endpoint 14

Offset: 0x1700 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 14. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 14.

23.9.6.44 USB2_QH_USB2D_QH_EP_14_IN_0

USB2D Queue Head for IN Endpoint 14

Offset: 0x1740 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 14. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 14.

23.9.6.45 USB2_QH_USB2D_QH_EP_15_OUT_0

USB2D Queue Head for OUT Endpoint 15

Offset: 0x1780 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 15. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 15.

23.9.6.46 USB2_QH_USB2D_QH_EP_15_IN_0

USB2D Queue Head for IN Endpoint 15

Offset: 0x17c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 15. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 15.

23.9.7 USB3 Controller Registers

23.9.7.1 USB2_CONTROLLER_2_USB2D_ID_0

USB2D Identification Register

Offset: 0x0 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:29	X	CIVERSION: Identifies the CI version
28:25	X	VERSION: Identifies the version of the core
24:21	X	REVISION: Revision number of the USB controller. This is set to 0x0.
20:16	X	TAG: Identifies the tag of the core
15:8	X	NID: One's complement version of ID. This field is set to 0xF9.
7:0	X	ID: Configuration number. This field is set to 0x06

23.9.7.2 USB2_CONTROLLER_2_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 0x8 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
3:1	X	NPORT: VUSB_HS_NUM_PORT-1: This host controller has only 1 port. So this field will always be 0.
0	X	HC: VUSB_HS_HOST: Indicates support for host mode. Set to 1.

23.9.7.3 USB2_CONTROLLER_2_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

Offset: 0xc | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: Number of endpoints supported by this device controller. Set to 16. This includes control endpoint 0.
0	X	DC: Device capable: Set to 1 indicating support for device mode.

23.9.7.4 USB2_CONTROLLER_2_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 0x10 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total number of address bits for the transmit buffer of each transmit endpoint. Set to 7. Each transmit buffer is 128 words deep.
15:8	X	TXADD: VUSB_HS_TX_ADD: Total number of address bits for the transmit buffer. Set to 11. The total depth of the transmit buffer is 2048 words.
7:0	X	TCBURST: VUSB_HS_TX_BURST: Maximum burst size supported by the transmit endpoints for data transfers. Set to 8.

23.9.7.5 USB2_CONTROLLER_2_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 0x14 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total number of address bits for the receive buffer. Set to 7. The total depth of the receive buffer is 128 words.
7:0	X	RXBURST: VUSB_HS_RX_BURST: Maximum burst size supported by the receive endpoints for data transfers. Set to 8.

23.9.7.6 USB2_CONTROLLER_2_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

Offset: 0x100 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
7:0	X	CAPLENGTH: Indicates which offset to add to the register base address at the beginning of the Operational Register. Set to 0x40.

23.9.7.7 USB2_CONTROLLER_2_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 0x102 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	HCIVERSION: Contains a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision. This host controller supports EHCI revision 1.00.

23.9.7.8 USB2_CONTROLLER_2_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

Offset: 0x104 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
27:24	X	N_TT: Number of Transaction Translators: Indicates the number of embedded transaction translators associated with the USB2.0 host controller. This field is always set to 1 indicating only 1 embedded TT is implemented in this implementation. This is a non-EHCI field to support embedded TT.
23:20	X	N_PTT: Number of Ports per Transaction Translator: Indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. Field always equals N_PORTS. This is a non-EHCI field to support embedded TT.
15:12	X	N_CC: Number of Companion Controller: Indicates the number of companion controllers. This field is set to 0.
11:8	X	N_PCC: Number of Ports per Companion Controller: Indicates the number of ports supported per internal companion controller. This field is set to 0.
4	X	PPC: Port Power Control: Indicates whether the host controller implementation includes port power control. 1 = Ports have port power switches 0= Ports do not have port power switches. This field affects the functionality of the port Power field in each port status and control register. This field is set to 1.
3:0	X	N_PORTS: Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. This field is fixed to 1, since this host controller only supports 1 port.

23.9.7.9 USB2_CONTROLLER_2_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

Offset: 0x108 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18	X	PPC: Per-Port Change Event Capability - RO. Default = 1b. This field indicates the support for per-port change events. This field is related to the USBCMD PPE field, USBSTS PPCI field, and USBINTR PPCE field.
17	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the USBCMD HIRD field, POSTSCx SSTS and DA fields, and HOSTPCx LEN, BA, and EPLPM fields.
15:8	X	EECP: EHCI Extended Capabilities Pointer: Indicates a capabilities list exists. A value of 00h indicates no extended capabilities are implemented. For this implementation this field is always "0".
7:4	X	IST: Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field will always be "0".
2	X	ASP: Asynchronous Schedule Park Capability. 1 = (Default) the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register. This field is always 1.
1	X	PFL: Programmable Frame List Flag. 0 = System software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. 1 = System software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This field will always be "1".

23.9.7.10 USB2_CONTROLLER_2_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

Offset: 0x120 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	DCIVERSION: The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

23.9.7.11 USB2_CONTROLLER_2_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 0x124 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
8	X	HC: Host Capable: 1 = This controller is capable of operating as an EHCI compatible USB 2.0 host controller. This field is set to 1.
7	X	DC: Device Capable: 1 = Controller is capable of operating as USB 2.0 device. This field is set to 1.

Bit	Reset	Description
5	X	LEN: Link Power Management Capability - RO. Default = 1b. This field indicates the support for LPM L1 state. This field is related to the DEVLCx ASUS, STL, BA, and NYT fields.
4:0	X	DEN: Device Endpoint Number: Number of endpoints built into the device controller. This is set to 16.

23.9.7.12 USB2_CONTROLLER_2_USB2D_EXTSTS_0

USB2D EXTSTS Register

Offset: 0x128 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000x

Bit	R/W	Reset	Description
4	RW	0x0	TI1: General Purpose Timer Interrupt 1 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
3	RW	0x0	TI0: General Purpose Timer Interrupt 0 - RWC. Default = 0b. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero. Writing a one to this bit will clear it.
2	RW	0x0	UPA: USB Host Periodic Interrupt (USBHSTPERINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
1	RW	0x0	UAI: USB Host Asynchronous Interrupt (USBHSTASYNCINT) R/WC. This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be zero. 0 = DISABLE 1 = ENABLE
0	RO	X	NAKI: NAK Interrupt Bit Read Only. This bit is read only. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared. 0 = DISABLE 1 = ENABLE

23.9.7.13 USB2_CONTROLLER_2_USB2D_USBEXTINTR_0

USB2D EXTINTR Register

Offset: 0x12c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000

Bit	Reset	Description
4	0x0	TIE1: General Purpose Timer Interrupt Enable 1 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
3	0x0	TIE0: General Purpose Timer Interrupt Enable 0 - RWC. Default = 0b. When this bit is a one, and the TI1 bit in the EXTSTS register is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the TI1 bit.
2	0x0	UPIE: UPIE Interrupt Enable. 1 = USB controller issues an interrupt if UPA bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	UAIE: UAIE Interrupt Enable. 1 = USB controller issues an interrupt if UAI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE
0	0x0	NAKE: NAK Interrupt Enable. 1 = USB controller issues an interrupt if NAKI bit in USBSTS register transitions. 0 = DISABLE 1 = ENABLE

23.9.7.14 USB2_CONTROLLER_2_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 0x130 | Read/Write: R/W | Reset: 0bxxxx00000000010000000x1x11x0000000

Bit	R/W	Reset	Description
27:24	RW	0x0	HIRD: Host Initiated Resume Duration RW. Default = 0000b. This has the same behavior as bits 7:4 of the BA field of the HOSTPCx register. When writing to this field all BA[7:4] fields of all HOSTPCx registers will be set to this value. This field is used by system software to specify the minimum amount of time the host controller will drive the K-state during a host-initiated resume from an LPM state (e.g., L1), and is conveyed to each LPM-enabled device (via the HIRD bits within an LPM Tokens bmAttributes field) upon entry into a low-power state. Note the host controller is required to drive resume signaling for at least the amount of time specified in the HIRD value conveyed to the device during any proceeding host-initiated resume. Also note that the host controller is not required to observe this requirement during device-initiated resumes. Encoding for this field is identical to the definition for the similarly named HIRD field within an LPM Token, specifically: a value 0000b equals 50 µs and each additional increment adds 75 µs. For example, the value 0001b equals 125 µs, and a value 1111b equals 1,175 µs (~1.2ms).
23:16	RW	0x8	ITC: Interrupt Threshold Control .Read/Write. Default 08h. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 00h = Immediate (no threshold) 01h = 1 micro-frame 02h = 2 micro-frames 04h = 4 micro-frames 08h = 8 micro-frames 10h = 16 micro-frames 20h = 32 micro-frames 40h = 64 micro-frames 0 = IMMEDIATE 2 = ONE_MF 4 = TWO_MF 8 = EIGHT_MF 16 = SIXTEEN_MF 32 = THIRTY_TWO_MF 64 = SIXTY_FOUR_MF
15	RW	0x0	FS2: Bit 2 of Frame List Size.
14	RW	0x0	ATDTW: Add DTD Tripwire. This bit is used as a semaphore when a dTD is added to an active (primed) endpoint. This bit is set and cleared by software and will be cleared by hardware when a hazard exists such that adding a dTD to a primed endpoint may go unnoticed. 0 = CLEAR 1 = SET
13	RW	0x0	SUTW: Setup Tripwire. This bit is used as a semaphore when the 8 bytes of setup data read extracted by the firmware. If the setup lockout mode is off, then there exists a hazard when new setup data arrives and firmware is copying setup data from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists. 0 = CLEAR 1 = SET

Bit	R/W	Reset	Description
11	RW	0x1	ASPE: Asynchronous Schedule Park mode Enable. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. This field is set to "1" in this implementation. 0 = DISABLE 1 = ENABLE
9:8	RW	0x3	ASP1_ASP0: Asynchronous Schedule Park Mode Count (OPTIONAL) Read/Write. If the Asynchronous Park Capability bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is RO. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when Park Mode Enable is a one as this will result in undefined behavior. This field is set to 3h in this implementation.
7	RO	X	LR: Light Host/Device Controller Reset (OPTIONAL). Not Implemented. This field will always be "0".
6	RW	0x0	IAA: Interrupt on Async Advance Doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in Host mode. Writing a one to this bit when device mode is selected will have undefined results. 0 = CLEAR 1 = SET
5	RW	0x0	ASE: Asynchronous Schedule Enable. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
4	RW	0x0	PSE: Periodic Schedule Enable. This bit controls whether the host controller skips processing the Periodic Schedule. 0 = Do not process the Periodic Schedule 1 = Use the PERIODICLISTBASE register to access the Periodic Schedule. Only the host controller uses this bit. 0 = DISABLE 1 = ENABLE
3:2	RW	0x0	FS1_FS0: Frame List Size 000 = Default. This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. Hence this field is Read/Write for this implementation. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3, and 2. 000 = 1024 elements (4096 bytes) Default value 001 = 512 elements (2048 bytes) 010 = 256 elements (1024 bytes) 011 = 128 elements (512 bytes) 100 = 64 elements (256 bytes) 101 = 32 elements (128 bytes) 110 = 16 elements (64 bytes) 111 = 8 elements (32 bytes) Only the host controller uses this field.
1	RW	0x0	RST: Controller Reset. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host Controller: When software writes a one to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset

Bit	R/W	Reset	Description
			is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller results in undefined behavior. Device Controller: When software writes a one to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines, etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. Writing a one to this bit in device mode is not recommended. 0 = CLEAR 1 = SET
0	RW	0x0	RS: Run/Stop: Host Controller: When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HCHalted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the host controller is in the Halted state (i.e., HCHalted in the USBSTS register is a one). Device Controller: Writing a one to this bit will cause the device controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this will cause a detach event. 0 = STOP 1 = RUN

23.9.7.15 USB2_CONTROLLER_2_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 0x134 | Read/Write: R/W | Reset: 0b00000000000000000x100x0000x0000

Bit	R/W	Reset	Description
31:16	RW	0x0	PPCI: Port-n Change Detect - RW. Default = 0000h. The definition for each bit is identical to the Port Change Detect field (bit 2 of this register) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set to a one then a port change event was detected on Port 2. The N_PORTS field in HCSPARAMS specifies how many ports are exposed by the host controller and thus how many bits in this field are valid.
15	RW	0x0	AS: Asynchronous Schedule Status. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. If AS = ASE: 1= Enable Asynchronous Schedule 0= Disable Asynchronous Schedule Only used by the host controller. 0 = DISABLE 1 = ENABLE
14	RW	0x0	PS: Periodic Schedule Status. This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. If PS = PSE: 1 = Periodic Schedule is enabled 0 = Periodic Schedule is disabled Only used by the host controller. 0 = DISABLE 1 = ENABLE
13	RO	X	RCL: Reclamation. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
12	RW	0x1	HCH: HCHalted. 1 = Default. This bit is a zero whenever the Run/Stop bit is a one. The Host Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g., internal error). Only used by the host controller. 0 = UNHALTED 1 = HALTED
11	RW	0x0	UALT_INT: ULPI alt_int Interrupt. 0 = Default. This interrupt bit is set when an RXCMD is received through the ULPI interface with bit 7 set (alt_int). The alt_int itself is set when an unmasked event occurs on any bit in the Carkit Interrupt Latch Register, in the ULPI PHY. The software should read the Carkit Interrupt Latch Register (Read to Clear) through the ULPI Viewport to check the source of the interrupt. Only present in designs where configuration constant VUSB_HS_PHY_ULPI = 1. 0 = NOT_ULPI_ALT_INT 1 = ULPI_ALT_INT
10	RW	0x0	ULPI_INT: ULPI Interrupt. This bit is set whenever an interrupt is received from ULPI PHY. Software writes 1 to clear it. 0 = NOT_ULPI_INT 1 = ULPI_INT
8	RW	0x0	SLI: DCSuspend. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller. 0 = NOTSUSPEND 1 = SUSPENDED
7	RW	0x0	SRI: SOF Received. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When an SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125 µs in HS mode and will be synchronized to the actual SOF that is received. Since device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to the connect and chirp. In host mode, this bit will be set every 125 µs and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit. 0 = SOF_NOT_RCVD 1 = SOF_RCVD
6	RW	0x0	URI: USB Reset Received. When the device controller detects a USB Reset and enters the default state, this bit is set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. 0 = NO_USB_RESET 1 = USB_RESET
5	RW	0x0	AAI: Interrupt and Asynchronous Advance. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller 0 = NOT_ADVANCED 1 = ADVANCED
4	RO	X	SEI: System Error. This bit is not used in this implementation and will always be set to "0". 0 = NO_ERROR 1 = ERROR
3	RW	0x0	FRI: Frame List Rollover. The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [1 3] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FHINDEX [12] toggles. Only used by the host controller. 0 = NO_ROLLOVER 1 = ROLLOVER

Bit	R/W	Reset	Description
2	RW	0x0	PCI: Port Change Detect. The Host Controller sets this bit to a 1 when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operational states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively. This bit is not EHCI compatible. 0 = NO_PORT_CHANGE 1 = PORT_CHANGE
1	RW	0x0	UEI: USB Error Interrupt. This bit gets set by the Host/Device controller when completion of a USB transaction results in an error condition. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also ad its interrupt on complete (IOC) bit set. 0 = NO_ERROR 1 = ERROR
0	RW	0x0	UI: USB Interrupt. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) as an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes. 0 = NO_INT 1 = INT

23.9.7.16 USB2_CONTROLLER_2_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 0x138 | Read/Write: R/W | Reset: 0b00000000000000000xxxx00x000000000

Bit	Reset	Description
31:16	0x0	PPCE: Port-n Change Detect Enable - RW. Default = 0000h. The definition for each bit in this field is identical to bit 2 of this register (Port Change Interrupt Enable) except these bits are specific to a given port, where bit 16 = Port 1, 17 = Port 2, etc. For example, if bit 17 is set (1b) then a port change event was detected on Port 2. When a bit in this field is a one, and the corresponding Port-n Change Detect bit in the USBSTS register is a one, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port-n Change Detect bit.
11	0x0	UALTIE: ULPI alt_int Interrupt Enable. 1 = USB controller issues an interrupt if ULPI_ALT_INT bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_ALT_INT bit. 0 = DISABLE 1 = ENABLE
10	0x0	ULPIE: ULPI Interrupt Enable. 1 = USB controller issues an interrupt if ULPI_INT bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the ULPI_INT bit. 0 = DISABLE 1 = ENABLE
8	0x0	SLE: Sleep Enable. 1 = Device controller issues an interrupt if DCSuspend bit in USBSTS register transitions. The interrupt is acknowledged by software by writing a 1 to the DCSuspend bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE
7	0x0	SRE: SOF Received Enable. 1 = Device controller issues an interrupt if SOF Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the SOF Received bit. 0 = DISABLE 1 = ENABLE
6	0x0	URE: USB Reset Enable.1 = Device controller issues an interrupt if USB Reset Received bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	AAE: Interrupt on Asynchronous Advance Enable. 1 = the host controller issues an interrupt at the next interrupt threshold if Interrupt on Async Advance bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
4	0x0	SEE: System Error Enable. 1 = Host/device controller issues an interrupt if the System Error bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the System Error bit. 0 = DISABLE 1 = ENABLE
3	0x0	FRE: Frame List Rollover Enable. 1 = Host controller issues an interrupt if Frame List Rollover bit in the USBSTS register = 1. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller. 0 = DISABLE 1 = ENABLE
2	0x0	PCE: Port Change Detect Enable. 1 = Host/device controller issues an interrupt if Port Change Detect bit in USBSTS register = 1. The interrupt is acknowledged by software clearing the Port Change Detect bit. 0 = DISABLE 1 = ENABLE
1	0x0	UEE: USB Error Interrupt Enable. 1 = Host controller issues an interrupt at the next interrupt threshold if the USBERRINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register. 0 = DISABLE 1 = ENABLE
0	0x0	UE: USB Interrupt Enable. 1 = Host/device issues an interrupt at the next interrupt threshold if the USBINT bit in USBSTS = 1. The interrupt is acknowledged by software clearing the USBINT bit. 0 = DISABLE 1 = ENABLE

23.9.7.17 USB2_CONTROLLER_2_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 0x13c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
13:0	X	FRINDEX: Frame Index. The value in this register increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. Each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N 000b (1024) 12 001b (512) 11 010b (256) 10 011b (128) 9 100b (64) 8 101b (32) 7 110b (16) 6 111b (8) 5 In device mode, the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode, bits 2:0 indicate the current micro-frame.

23.9.7.18 USB2_CONTROLLER_2_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 0x144 | Read/Write: R/W | Reset: 0b00000000000000000000xxxxxxxxxxxx

Bit	Reset	Description
31:25	0x0	USBADR: Device mode. The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS request.
24	0x0	USBADRA: Device Address Advance. Default=0. When this bit is 0, any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD cannot write of the device address within 2 ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
31:12	0x0	BASEADR: Host mode: This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.

23.9.7.19 USB2_CONTROLLER_2_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 0x148 | Read/Write: R/W | Reset: 0b00000000000000000000000000xxxxx

Bit	Reset	Description
31:11	0x0	EPBASE: Device mode. This register contains the address of the top of the endpoint list in system memory. These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). Only used by the device controller.
31:5	0x0	ASYBASE: Host mode. This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.

23.9.7.20 USB2_CONTROLLER_2_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 0x14c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0x

Bit	R/W	Reset	Description
1	RW	0x0	TTAC: Embedded TT Async Buffers Clear. (Read/Write to set). This field will clear all pending transactions in the embedded TT Async Buffer(s). The clear will take as much time as necessary to clear buffer without interfering with a transaction in progress. TTAC will return to zero after being set by software only after the actual clear occurs.
0	RO	X	TTAS: Embedded TT Async Buffers Status. (Read Only). This read only bit will be 1 if one or more transactions are being held in the embedded TT Async. Buffers. When this bit is a zero, then all outstanding transactions in the embedded TT have been flushed.

23.9.7.21 USB2_CONTROLLER_2_USB2D_BURSTSIZE_0

USB2D Burst Size Register

Offset: 0x150 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000100000001000

Bit	Reset	Description
15:8	0x8	TXPBURST: Programmable TX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus.
7:0	0x8	RXPBURST: Programmable RX Burst Length. (Read/Write). This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory.

23.9.7.22 USB2_CONTROLLER_2_USB2D_TXFILLTUNING_0

USB2D Transmit Fill Tuning Register

Offset: 0x154 | Read/Write: R/W | Reset: 0bxxxxxxxxxx000010xxx0000000000000

Bit	Reset	Description
21:16	0x2	TXFIFOTHRES: FIFO Burst Threshold. (Read/Write). This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
12:8	0x0	TXSCHHEALTH: Scheduler Health Counter. (Read/Write To Clear) [Default = 0]. This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max out at 31.
7:0	0x0	TXSCHOH: Scheduler Overhead. (Read/Write) [Default = 0]. This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267 µs when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333 µs when a device is connected in Low/Full Speed Mode.

23.9.7.23 USB2_CONTROLLER_2_USB2D_ICUSB_CTRL_0

USB2D ICUSB Control Register

This register enables and controls the ICUSB FS/LS transceiver.

Offset: 0x15c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000

Bit	Reset	Description
3	0x0	IC_ENB1: ICUSB transceiver enable. This bit enables the ICUSB transceiver. To enable the interface, the bits PTS must be set to 11 in the PORTSCx. Writing a '1' to this bit selects the IC_USB interface. 0 = DISABLE 1 = ENABLE
2:0	0x0	IC_VDD1: ICUSB voltage select. It selects which voltage is being supplied to the ICUSB peripheral. 000 = No voltage 001 = 1.0V – reserved 010 = 1.2V - reserved 011 = 1.5V - reserved 100 = 1.8V 101 = 3.0V

Bit	Reset	Description
		110 = reserved 111 = reserved The Voltage negotiation should happen between enabling port power (PP) and asserting the run/stop bit in register.

23.9.7.24 USB2_CONTROLLER_2_USB2D_ULPI_VIEWPORT_0

USB2D ULPI Viewport Register

This register provides indirect access to the ULPI PHY register set. Although the USB controller performs access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access.

Note: WRITES TO THE ULPI THROUGH THE VIEWPORT CAN SUBSTANTIALLY HARM STANDARD USB OPERATIONS. CURRENTLY NO USAGE MODEL HAS BEEN DEFINED WHERE SOFTWARE SHOULD NEED TO EXECUTE WRITES DIRECTLY TO THE ULPI PHY. SEE EXCEPTION REGARDING OPTIONAL FEATURES BELOW.

Note: EXECUTING READ OPERATIONS THROUGH THE ULPI VIEWPORT SHOULD HAVE NO HARMFUL SIDE EFFECTS TO STANDARD USB OPERATIONS.

There are two operations that can be performed with the ULPI Viewport: wakeup and read /write operations.

The wakeup operation is used to put the ULPI interface into normal operation mode and re-enable the clock, if necessary. A wakeup operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or Carkit mode.

The ULPI state can be determined by reading the sync state bit (ULPI_SYNC_STATE). If this bit is a one, then ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPI_SYNC_STATE indicates a 0 then then read/write operations will not be able to execute. Undefined behavior will result if ULPI_SYNC_STATE = 0 and a read or write operation is performed.

To execute a wakeup operation, write all 32 bits of the ULPI Viewport where ULPI_PORT is constructed appropriately and the ULPI_WAKEUP bit is a 1 and ULPI_RUN bit is a 0. Poll the ULPI Viewport until ULPI_WAKEUP is zero for the operation to complete.

To execute a read or write operation, write all 32 bits of the ULPI Viewport where ULPI_DATA_WR, ULPI_REG_ADDR, ULPI_PORT, ULPI_RD_WR are constructed appropriately and the ULPI_RUN bit is a 1. Poll the ULPI Viewport until ULPI_RUN is zero for the operation to complete. Once ULPI_RUN is zero, the ULPI_DATA_RD will be valid if the operation was a read.

The polling method above can be changed to interrupt driven using the ULPI interrupt defined in the USBSTS and USBINTR registers. When a wakeup or read/write operation is complete, the ULPI_INT interrupt will be set.

There are several optional features that may need to be enabled or disabled by system software as part of system configuration. These bits are contained in the Interface and OTG Control registers of the ULPI PHY register set. These registers also contain bits which are controlled by the link dynamically and therefore should be only modified by system software using the Set/Clear access method. Direct writes to these registers could have harmful side effects to the standard USB operations. The optional bits are as follows: Bits 3 through 7 in the Interface Control register and Bits 6 and 7 in the OTG Control register.

Refer to the ULPI Specification Revision 1.1 for further information on the use of the optional features.

Offset: 0x160 | Read/Write: R/W | Reset: 0b000xx00000000000xxxxxxxx00000000

Bit	R/W	Reset	Description
31	RW	0x0	ULPI_WAKEUP: ULPI Wakeup. Writing the 1 to this bit will begin the wakeup operation. The bit will automatically transition to 0 after the wakeup is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
30	RW	0x0	ULPI_RUN: ULPI read/write Run. Writing the 1 to this bit will begin the read/write operation. The bit will automatically transition to 0 after the read/write is complete. Once this bit is set, the driver cannot set it back to 0. Note: The driver must never execute a wakeup and a read/write operation at the same time. 0 = CLEAR 1 = SET
29	RW	0x0	ULPI_RD_WR: ULPI read/write control. (0) Read; (1) Write. This bit selects between running a read or write operation. 0 = READ 1 = WRITE
27	RO	X	ULPI_SYNC_STATE: ULPI sync state. (1) Normal Sync. State. (0) In another state (i.e., Carkit, serial, low power). This bit represents the state of the ULPI interface. 0 = NOT_NORMAL 1 = NORMAL
26:24	RW	0x0	ULPI_PORT: ULPI PHY port number. This field should be always written as 0.
23:16	RW	0x0	ULPI_REG_ADDR: ULPI PHY register address. When doing a read or write operation to the ULPI PHY, the address of the ULPI PHY register being accessed is written to this field.
15:8	RO	X	ULPI_DATA_RD: ULPI PHY data read. The data from the ULPI PHY register can be read from here after the read operation completes.
7:0	RW	0x0	ULPI_DATA_WR: ULPI PHY data write. The data to write to the ULPI PHY register is written here.

23.9.7.25 USB2_CONTROLLER_2_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 0x174 | Read/Write: R/W | Reset: 0b0000000xx0000000xxx1xx00x0xx010x

Bit	R/W	Reset	Description
31:25	RW	0x0	DA: Device Address. Default = 0000000b. The 7-bit USB device address for the device attached to an immediately downstream of the associated root port. A value of zero indicates no device is present or software support for this feature is not present. This is used by the Controller when sending the LPM token. This field is only valid when the core is operating in host mode. If in device mode, it will be read only and always equal to 0000000b.

Bit	R/W	Reset	Description
24:23	RO	X	SSTS: Suspend Status. Default = 00b. These two bits are used by software to determine whether an L1-based suspend request was successful, specifically: 00b - L1 state entered with success. ACK received from peripheral. 01b - NYET received from peripheral. It was not able to enter L1 state this time. 10b - L1 state not supported by peripheral. STALL received. 11b - Peripheral did not respond or an error occurred. The value of this field is only valid when the port resides in the L0 state - that is, the meaning of these bits is invalid whenever bit 7 of this register (SUSP) is one. Ideally, the Controller driver should read this register if it receives an interrupt after issuing a suspend using L1 support. In case of a non-success a port change interrupt will be fired and this field should be checked for a possible L1 failure. This field is only valid when the core is operating in host mode. If in device mode it will be always equal to 00b. 0 = L1STATE_ENTERED 1 = NYET_PERIPH 2 = L1STATE_NOT_SUPPORTED 3 = PERIPH_NORESP_ERR
22	RW	0x0	WKOC: Default = 0b. Wake on Over-current Enable: Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power (PP) is zero. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. 0 = DISABLE 1 = ENABLE
21	RW	0x0	WKDS: Wake on Disconnect Enable: Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power (PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. This bit should not be written to 1 if there is no device connected. After the device disconnect is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
20	RW	0x0	WKCN: Wake on Connect Enable: Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power (PP) is zero or in device mode. This bit should only be used when operating in Host mode. Writing this bit to 1 while the controller is working in device mode can result in undefined behavior. This bit should not be written to 1 while the device is connected. After the device connection is detected, this bit should be cleared to 0. 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any value other than zero indicates that the port is operating in test mode. Value Specific Test. 0000b: Not enabled. 0001b: J_ STATE. 0010b: K_STATE. 0011b: SEQ_NAK. 0100b: Packet. 0101b: FORCE_ENABLE. 0110b to 1111b: Reserved. Refer to Chapter 7 of the USB Specification Revision 2.0 for details on each test mode. 0 = NORMAL_OP 1 = TEST_J 2 = TEST_K 3 = TEST_SE0_NAK 4 = TEST_PKT 5 = TEST_FORCE_ENABLE
15:14	RO	X	PIC: Port Indicator Control: This field is not supported in the current implementation. Please use a GPIO if you wish to use Port Indicators.
13	RO	X	PO: Port Owner. Port owner handoff is not implemented in this design, therefore this bit will always be 0.

Bit	R/W	Reset	Description
12	RW	0x1	PP: Port Power: The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0b 0b Read Only. A device controller with no OTG capability does not have port power control switches. 1b 1b/0b RW. Host/OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e., PP equals a 0), the port is non-functional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitioned by the host controller driver from a one to a zero (removing power from the port). 0 = NOT_POWERED 1 = POWERED
11:10	RO	X	LS: Line state. These bits reflect the current logical levels of the D+ (bit 10) and D- (bit 11) signal lines. The encoding of the bits is: 00b = SE0 01b = J-state 10b = K-state 11b = Undefined The value of this field is undefined if Port Power (PP) is zero in host mode. In host mode, the use of line-state by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of line-state by the device controller driver is not necessary. 0 = SE0 1 = J_STATE 2 = K_STATE 3 = UNDEFINED
9	RW	0x0	SLP: Suspend using L1 - RW. Default = 0b. When this bit is set to '1' and a non-zero Device Address (DA) is specified Controller will instigate L1 entry during suspend (bit 7) and L1 exit during resume (bit 6). When set to zero the Controller will use the legacy (L2) mechanism. Software should only set this bit when the device attached immediately downstream supports L1 transitions. When acting as device, this bit is read-only and set to '1' by the hardware when the Controller enters is L1 state (LPM token received and accepted). Note: HSP is redundant with PSPD(27:26). This bit is not defined in the EHCI specification.
8	RW	0x0	PR: This field is zero if Port Power (PP) is zero. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver. In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. 0 = NOT_USB_RESET 1 = USB_RESET

Bit	R/W	Reset	Description
7	RO	X	SUSP: Port suspend. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the Force Port Resume bit to zero. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (i.e., Port enabled bit is a zero) the results are undefined. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: Read Only. This bit is a read only status bit. 0 = NOT_SUSPEND 1 = SUSPEND
6	RW	0x0	FPR: Force Port Resume. 1= Resume detected/driven on port. 0=No resume (K state) detected/driven on port. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit remains a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time the resume operation to clear the bit when the port control state switches to HS or FS idle. This field is zero if Port Power (PP) is zero in host mode. This bit is not-EHCI compatible. In Device mode: After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to one. Software should ensure that the PHY clock is operational before writing a 1 to this bit to start the resume sequence. This is true for both Device and Host modes. 0 = NO_RESUME 1 = RESUME
5	RO	X	OCC: Over-current Change: Not supported 0 = NO_CHANGE 1 = CHANGE
4	RO	X	OCA: Over-current Active: Not supported 0 = NO_OVER_CURRENT 1 = OVER_CURRENT
3	RW	0x0	PEC: Port Enable/Disable Change: 1=Port enabled/disabled status has changed. 0=No change. In Host Mode: For the root hub, this bit gets set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if Port Power (PP) is zero. In Device mode: The device port is always enabled. (This bit will be zero) 0 = NO_CHANGE 1 = CHANGE
2	RW	0x1	PE: Port Enabled/Disabled: 1=Enable. 0=Disable (default) In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b)

Bit	R/W	Reset	Description
			downstream propagation of data is blocked except for reset. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: The device port is always enabled. (This bit will be one) 0 = PORT_DISABLED 1 = PORT_ENABLED
1	RW	0x0	CSC: Connect Status Change: 1 =Change in Current Connect Status. 0=No change (default). In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a one to it. This field is zero if Port Power (PP) is zero in host mode. This bit is undefined in device controller mode. 0 = NO_CHANGE 1 = CHANGE
0	RO	X	CCS: Current Connect Status: In Host Mode: 1=Device is present on port. 0=No device is present (default). This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is zero if Port Power (PP) is zero in host mode. In Device Mode: 1=Attached 0=Not Attached (default) A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended. 0 = NOT_CONNECTED 1 = CONNECTED

23.9.7.26 USB2_CONTROLLER_2_USB2D_HOSTPC1_DEVLC_0

USB2D Device Mode LPM Behavior and Control Register

This register controls the LPM behavior and controls the behavior of each USB port in device mode only. This register shares the same address with the HOSTPC1 register (which is used only in host mode).

USB2D Host Mode LPM Behavior and Control Register

This register controls the LPM behavior and controls the behavior of each USB port in host mode only. This register shares the same address with the DEVLC register (which is used only in device mode).

Offset: 0x1b4 | Read/Write: R/W | Reset: 0b0000xxx0000000000000xxxxxxxxxxx0

Bit	R/W	Reset	Description
31:29	RW	0x0	PTS: Parallel transceiver select. This bit is not defined in the EHCI specification. 0 = UTMI 1 = RESERVED 2 = ULPI 3 = ICUSB_SER 4 = HSIC
28	RW	0x0	STS: Serial transceiver not selected. This is the only value supported. This bit is not defined in the EHCI specification. 0 = PARALLEL_IF 1 = SERIAL_IF
27	RO	X	PTW: Parallel Transceiver Width. Fixed to 0. This bit is not defined in the EHCI specification. 0 = EIGHT_BIT 1 = RESERVED

Bit	R/W	Reset	Description
26:25	RO	X	PSPD: This register field indicates the speed at which the port is operating. 00 = Full Speed 01 = Low Speed 10 = High Speed. This bit is not defined in the EHCI specification. 0 = FULL_SPEED 1 = LOW_SPEED 2 = HIGH_SPEED 3 = RESERVED
24	RW	0x0	ALPD: Auto Low Power While Disconnect - RW. Default = 0b. If set, this feature will be enabled and every time the port enters the disconnect state it will also enter in low power state, disabling the transceiver clock. The behavior will be same as if the PHCD bit was enabled(in fact this bit will be set to 1 as soon as low power mode is enabled).When this field is set the WKCN field of PORTSCx register(Wake on Connect Enable) will also be set. This way the core will wake up in case a connect is detected. There will be a delay between the detection of a disconnect and actually enter in low power mode. This delay can be controlled by writing to the ALPDD field in the USBMODE register. 0 = DONT_AUTO_LOW_POWER_WHILE_DISCONNECT 1 = AUTO_LOW_POWER_WHILE_DISCONNECT
23	RW	0x0	PFSC: Port Force Full Speed Connect - RW. Default = 0b. Writing this bit to a '1b' will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as High Speed. This is useful for testing FS configurations with an HS host, hub or device. This bit is not defined in the EHCI specification. 0 = DONT_FORCE_FULL_SPEED 1 = FORCE_FULL_SPEED
22	RW	0x0	PHCD: PHY Low Power Suspend - Clock disable: Writing this bit to a 1 will disable the PHY clock. Write a 0 enables it. Reading this bit will indicate the status of the PHY clock. NOTE: The PHY clock cannot be disabled if it is being used as the system clock. In device mode, the PHY can be put into Low Power Clock Disable when the device is not running (USBCMD RS=0b) or the host has signaled suspend (PORTSCx SUSP=1b). Low Power Clock Disable will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the Controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend Clock Disable when the downstream device has been put into suspend mode or when no downstream device is connected. Low Power Clock Disable is completely under the control of software. 0 = DISABLE 1 = ENABLE
21:20	RW	0x0	LPMX: Auto LPM set. Default = 00b. This bit field is valid during Host Mode Only. For Device Mode, this is reserved. Value Meaning 00b Disables auto LPM. 01b If there is no activity for a certain number of SOFs the controller will send an LPM token, enter in suspend and issue a port change interrupt. 10b Same as above but without issuing an interrupt. 11b Reserved for futures LPM enhancements. The detection of no activity is based on the absence of response from the downstream device. Because of this limitation this feature should not be used if ISO OUT endpoints are being used (as no handshake is expected).
17	RW	0x0	ASUS: Auto Low Power. Default = 0b. This bit field is valid during Device Mode Only. In Host Mode it is part of ELPM field. This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters the suspend state, it will also enter the low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	EPLPM: Endpoint for LPM token. Default = 0000b. This bit field is valid during Host Mode only. For Device Mode, bits [19:18] are reserved and bit 17 is ASUS, while bit 16 is STL. This sets the endpoint number to which the LPM token will be sent. It will be directly mapped to the ENDP field in the LPM token with the EXT PID.

Bit	R/W	Reset	Description
16	RW	0x0	STL: STALL reply to LPM token. Default = 0b. This bit field is valid during Device Mode only. In Host Mode it is a part of ELPM field. When this bit is set to '1', the Controller will reply always with STALL to all incoming LPM tokens. This bit overrides the LPM NYET bit (NYT). 0 = DISABLE 1 = ENABLE
15:12	RW	0x0	LPMFRM: Auto LPM SOF Threshold. Default = 0000b. This bit field is valid during Host Mode only. For Device Mode, this is reserved. This holds the SOF counter threshold. When the number of SOF with no activity in between reaches this threshold and the auto LPM is enabled, an LPM token will be sent and the port will enter in suspend state. The SOF counter for this threshold is incremented each 125 µs, even if the port is not in HS operation.
11:1	RO	X	BA: BmAttributes . Default = 00000000000b. This holds the bmAttributes field of the LPM sub-token received, after the EXT PID token.
0	RW	0x0	NYT_ASUS: NYET reply to LPM token. Default = 0b. This bit is NYT during Device Mode. Details follow: When this bit is to '1', the device controller will NYET all the LPM tokens. When this bit is set to '0', the Controller will ACK all the LPM tokens if the STALL bit (STL) is also set to '0'. This bit is ASUS in Host Mode. Details follow: This bit is used to control the auto low power feature. If set, the auto low power feature will be enabled and every time the port enters in suspend state it will also enter in low power state, disabling the transceiver clock. The behavior will be the same as if the PHCD bit was enabled (in fact this bit will be set to '1' as soon as low power mode is enabled). 0 = DISABLE 1 = ENABLE

23.9.7.27 USB2_CONTROLLER_2_USB2D_OTGSC_0

USB2D On-The-Go (OTG) Status and Control Register

Offset: 0x1f4 | Read/Write: R/W | Reset: 0bx0000000x0000000xxxxxxxxxx000x00

Bit	R/W	Reset	Description
30	RW	0x0	DPIE: Data Pulse Interrupt Enable. Setting this bit enables the Data pulse interrupt. 0 = DISABLE 1 = ENABLE
29	RW	0x0	ONEMSE: 1 millisecond timer Interrupt enable. Setting this bit enables the 1 millisecond timer interrupt. 0 = DISABLE 1 = ENABLE
28	RW	0x0	BSEIE: B Session End Interrupt Enable. Setting this bit enables the B session end interrupt 0 = DISABLE 1 = ENABLE
27	RW	0x0	BSVIE: B Session Valid Interrupt Enable. Setting this bit enables the B session valid interrupt 0 = DISABLE 1 = ENABLE
26	RW	0x0	ASVIE: A Session Valid Interrupt Enable. Setting this bit enables the A session valid interrupt 0 = DISABLE 1 = ENABLE
25	RW	0x0	AVVIE: A VBus Valid Interrupt Enable. Setting this bit enables the A VBus valid interrupt 0 = DISABLE 1 = ENABLE
24	RW	0x0	IDIE: USB ID Interrupt Enable. Setting this bit enables the USB ID interrupt 0 = DISABLE 1 = ENABLE
22	RW	0x0	DPIS: Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0).PortPower = Off (0). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET

Bit	R/W	Reset	Description
21	RW	0x0	ONEMSS: 1 millisecond timer Interrupt Status: This bit is set once every millisecond. Software writes a 1 to clear it. 0 = INT_CLEAR 1 = INT_SET
20	RW	0x0	BSEIS: B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software writes a 1 to clear this bit . 0 = INT_CLEAR 1 = INT_SET
19	RW	0x0	BSVIS: B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
18	RW	0x0	ASVIS: A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software writes a one to clear this bit. 0 = INT_CLEAR 1 = INT_SET
17	RW	0x0	AVVIS: A VBus Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
16	RW	0x0	IDIS: USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software writes a 1 to clear this bit. 0 = INT_CLEAR 1 = INT_SET
14	RO	X	DPS: Data Bus Pulsing Status. A 1 indicates data bus pulsing is being detected on the port. 0 = STS_CLEAR 1 = STS_SET
13	RO	X	ONEMST: 1 millisecond timer toggle. This bit toggles once per millisecond 0 = STS_CLEAR 1 = STS_SET
12	RO	X	BSE: B session End. Indicates VBus is below the B session end threshold 0 = STS_CLEAR 1 = STS_SET
11	RO	X	BSV: B Session Valid. Indicates VBus is above the B session valid threshold 0 = STS_CLEAR 1 = STS_SET
10	RO	X	ASV: A Session Valid. Indicates VBus is above the A session valid threshold 0 = STS_CLEAR 1 = STS_SET
9	RO	X	AVV: A VBus Valid. Indicates VBus is above the A VBus valid threshold 0 = STS_CLEAR 1 = STS_SET
8	RO	X	ID: USB ID: 0 = A-device 1 = B-device 0 = A_DEV 1 = B_DEV
5	RW	0x0	IDPU: USB ID Pullup 0 = CLEAR 1 = SET
4	RW	0x0	DP: Data Pulsing. Setting this bit causes the pull-up on DP to be asserted for data pulsing during SRP. 0 = NO_DATA_PULSE 1 = DATA_PULSE
3	RW	0x0	OT: OTG Termination. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM. 0 = NO_OTG_TERM 1 = OTG_TERM

Bit	R/W	Reset	Description
1	RW	0x0	VC: VBUS Charge. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP. 0 = NO_VBUS_CHRG 1 = VBUS_CHRG
0	RW	0x0	VD: VBUS_Discharge. Read/write. Setting this bit causes Vbus to discharge through a resistor. 0 = NO_VBUS_DISCHRG 1 = VBUS_DISCHRG

23.9.7.28 USB2_CONTROLLER_2_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 0x1f8 | Read/Write: R/W | Reset: 0b00000000000000000xxxxxxxxx0xxxxx

Bit	R/W	Reset	Description
31:16	RW	0x0	ALPDD: Auto Low Power While Disconnect Delay, Used when the ALPD field of the HOSTPCx register is set. Defines the delay between disconnect detection and entering in low power mode. The delay is <this register value>*64*100/3 in milliseconds. The maximum value is 65535*64*100/3 = 139,808 ms. The minimum value is 0. Only used in host mode.
15	RW	0x0	SRT: Shorten USB Reset Time. Software should never set this bit to 1.
5	RW	0x0	VBPS: VBUS Power Select This can be used by logic that selects between an on-chip Vbus power source (charge pump) and an off-chip source in systems when both are available. Only to be used in Host Mode. No functionality is implemented for this, so software should not use this.
4	RO	X	SDIS: Stream disable: 1 Streaming is disabled - helpful to avoid overruns/underruns when system load is too high. 0 = STREAM_ENABLE 1 = STREAM_DISABLE
3	RO	X	SLOM: Setup Lockout Mode: In device mode, this bit controls the behavior of the setup lockout mechanism. 0 - Setup lockout is ON (default) 1 - Setup lockout is OFF. Firmware requires the use of setup tripwire semaphore in USB2D_USBCMD register. 0 = LOCKOUT_ON 1 = LOCKOUT_OFF
2	RO	X	ES: Endian Select: Note: For this implementation, this should be always set to 0 (little endian). 0 = LITTLE_ENDIAN 1 = RESERVED
1:0	RO	X	CM: Controller Mode: The controller mode will default to an idle state and will need to be initialized to the desired operating mode after reset. This register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register. 00 = Idle [Default] 01 = Reserved 10 = Device Controller 11 = Host Controller. 0 = IDLE 1 = RESERVED 2 = DEVICE_MODE 3 = HOST_MODE

23.9.7.29 USB2_CONTROLLER_2_USB2D_ENDPTNAK_0

USB2D Endpoint NAK Register

Offset: 0x200 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTN: TX Endpoint NAK R/WC. Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. 0 = CLEAR 1 = SET
15:0	0x0	EPRN: RX Endpoint NAK R/WC. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. 0 = CLEAR 1 = SET

23.9.7.30 USB2_CONTROLLER_2_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable Register

Offset: 0x204 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	EPTNE: TX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE
15:0	0x0	EPRNE: RX Endpoint NAK Enable R/W. Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. 0 = DISABLE 1 = ENABLE

23.9.7.31 USB2_CONTROLLER_2_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 0x208 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15	0x0	ENDPTSETUPSTAT15: Endpoint 15 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
14	0x0	ENDPTSETUPSTAT14: Endpoint 14 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
13	0x0	ENDPTSETUPSTAT13: Endpoint 13 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
12	0x0	ENDPTSETUPSTAT12: Endpoint 12 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
11	0x0	ENDPTSETUPSTAT11: Endpoint 11 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
10	0x0	ENDPTSETUPSTAT10: Endpoint 10 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
9	0x0	ENDPTSETUPSTAT9: Endpoint 9 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
8	0x0	ENDPTSETUPSTAT8: Endpoint 8 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
7	0x0	ENDPTSETUPSTAT7: Endpoint 7 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
6	0x0	ENDPTSETUPSTAT6: Endpoint 6 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
5	0x0	ENDPTSETUPSTAT5: Endpoint 5 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

Bit	Reset	Description
4	0x0	ENDPTSETUPSTAT4: Endpoint 4 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
3	0x0	ENDPTSETUPSTAT3: Endpoint 3 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
2	0x0	ENDPTSETUPSTAT2: Endpoint 2 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
1	0x0	ENDPTSETUPSTAT1: Endpoint 1 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD
0	0x0	ENDPTSETUPSTAT0: Endpoint 0 Setup Status: For every setup transaction that is received, this bit is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to it after it has read the setup data from Queue head. The response to a setup packet (as in the order of operations and total response time) is crucial to limit bus time-outs while the setup lock-out mechanism is engaged. This register is only used in device mode. 0 = NOT_RCVD 1 = SETUP_RCVD

23.9.7.32 USB2_CONTROLLER_2_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 0x20c | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	PETB15: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
30	0x0	PETB14: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
29	0x0	PETB13: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
28	0x0	PETB12: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
27	0x0	PETB11: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
26	0x0	PETB10: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
25	0x0	PETB9: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
24	0x0	PETB8: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
23	0x0	PETB7: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
22	0x0	PETB6: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
21	0x0	PETB5: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
20	0x0	PETB4: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
19	0x0	PETB3: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
18	0x0	PETB2: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
17	0x0	PETB1: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
16	0x0	PETB0: Prime Endpoint Transmit Buffer: This bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction on this endpoint. Software should write a "1" to this bit when posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
15	0x0	PERB15: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
14	0x0	PERB14: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
13	0x0	PERB13: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
12	0x0	PERB12: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
11	0x0	PERB11: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
10	0x0	PERB10: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
9	0x0	PERB9: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
8	0x0	PERB8: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
7	0x0	PERB7: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
6	0x0	PERB6: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

Bit	Reset	Description
5	0x0	PERB5: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
4	0x0	PERB4: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
3	0x0	PERB3: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
2	0x0	PERB2: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
1	0x0	PERB1: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME
0	0x0	PERB0: Prime Endpoint Receive Buffer: This bit is used to request that a buffer prepared for a receive operation when a USB host initiates a USB OUT transaction to this endpoint. Software should write a one to this bit whenever posting a new transfer descriptor to this endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when this endpoint is successfully primed. This is only used in device mode. 0 = DONT_PRIME 1 = PRIME

23.9.7.33 USB2_CONTROLLER_2_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 0x210 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	FETB15: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
30	0x0	FETB14: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
29	0x0	FETB13: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
28	0x0	FETB12: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
27	0x0	FETB11: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
26	0x0	FETB10: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
25	0x0	FETB9: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
24	0x0	FETB8: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
23	0x0	FETB7: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
22	0x0	FETB6: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
21	0x0	FETB5: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
20	0x0	FETB4: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
19	0x0	FETB3: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
18	0x0	FETB2: Flush Endpoint Transmit Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
17	0x0	FETB1: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
16	0x0	FETB0: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
15	0x0	FERB15: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
14	0x0	FERB14: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
13	0x0	FERB13: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
12	0x0	FERB12: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
11	0x0	FERB11: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
10	0x0	FERB10: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
9	0x0	FERB9: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
8	0x0	FERB8: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
7	0x0	FERB7: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
6	0x0	FERB6: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
5	0x0	FERB5: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
4	0x0	FERB4: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
3	0x0	FERB3: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
2	0x0	FERB2: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH
1	0x0	FERB1: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

Bit	Reset	Description
0	0x0	FERB0: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear any primed buffers. If a packet is in progress for the associated endpoint that transfer will continue until completion. Hardware clears this register after the endpoint flush operation is successful. This is only used in device mode. 0 = DONT_FLUSH 1 = FLUSH

23.9.7.34 USB2_CONTROLLER_2_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 0x214 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	ETBR15: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
30	X	ETBR14: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
29	X	ETBR13: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
28	X	ETBR12: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
27	X	ETBR11: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
26	X	ETBR10: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
25	X	ETBR9: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
24	X	ETBR8: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
23	X	ETBR7: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
22	X	ETBR6: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
21	X	ETBR5: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
20	X	ETBR4: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
19	X	ETBR3: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
18	X	ETBR2: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
17	X	ETBR1: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
16	X	ETBR0: Endpoint Transmit Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
15	X	ERBR15: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
14	X	ERBR14: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
13	X	ERBR13: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
12	X	ERBR12: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
11	X	ERBR11: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
10	X	ERBR10: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
9	X	ERBR9: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
8	X	ERBR8: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
7	X	ERBR7: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
6	X	ERBR6: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
5	X	ERBR5: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

Bit	Reset	Description
4	X	ERBR4: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
3	X	ERBR3: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
2	X	ERBR2: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
1	X	ERBR1: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY
0	X	ERBR0: Endpoint Receive Buffer Ready: One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. This is only used in device mode. 0 = NOT_READY 1 = READY

23.9.7.35 USB2_CONTROLLER_2_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 0x218 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ETCE15: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
30	0x0	ETCE14: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
29	0x0	ETCE13: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
28	0x0	ETCE12: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
27	0x0	ETCE11: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
26	0x0	ETCE10: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
25	0x0	ETCE9: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
24	0x0	ETCE8: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
23	0x0	ETCE7: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
22	0x0	ETCE6: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
21	0x0	ETCE5: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
20	0x0	ETCE4: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
19	0x0	ETCE3: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
18	0x0	ETCE2: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
17	0x0	ETCE1: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
16	0x0	ETCE0: Endpoint Transmit Complete Event: Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
15	0x0	ERCE15: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
14	0x0	ERCE14: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
13	0x0	ERCE13: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
12	0x0	ERCE12: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
11	0x0	ERCE11: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
10	0x0	ERCE10: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

Bit	Reset	Description
9	0x0	ERCE9: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
8	0x0	ERCE8: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
7	0x0	ERCE7: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
6	0x0	ERCE6: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
5	0x0	ERCE5: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
4	0x0	ERCE4: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
3	0x0	ERCE3: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
2	0x0	ERCE2: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
1	0x0	ERCE1: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE
0	0x0	ERCE0: Endpoint Receive Complete Event: Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a one clears the corresponding bit in this register. This is only used in device mode. 0 = NOT_COMPLETE 1 = COMPLETE

23.9.7.36 USB2_CONTROLLER_2_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 0x21c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23	X	TXE: TX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
19:18	X	TXT: TX Endpoint Type. Endpoint0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
16	X	TXS: TX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL
7	X	RXE: RX Endpoint Enable. Endpoint 0 is always enabled. 0 = DISABLE 1 = ENABLE
3:2	X	RXT: RX Endpoint Type. Endpoint 0 is fixed as a Control Endpoint. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
0	X	RXS: RX Endpoint Stall: Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0 = EP_OK 1 = EP_STALL

23.9.7.37 USB2_CONTROLLER_2_USB2D_ENDPTCTRL1_0

USB2D Endpoint Control 1 Register

Offset: 0x220 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR

Bit	R/W	Reset	Description
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above, 0 = EP_OK 1 = EP_STALL

23.9.7.38 USB2_CONTROLLER_2_USB2D_ENDPTCTRL2_0

USB2D Endpoint Control 2 Register

Offset: 0x224 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ

Bit	R/W	Reset	Description
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.39 USB2_CONTROLLER_2_USB2D_ENDPTCTRL3_0

USB2D Endpoint Control 3 Register

Offset: 0x228 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ

Bit	R/W	Reset	Description
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.40 USB2_CONTROLLER_2_USB2D_ENDPTCTRL4_0

USB2D Endpoint Control 4 Register

Offset: 0x22c | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.41 USB2_CONTROLLER_2_USB2D_ENDPTCTRL5_0

USB2D Endpoint Control 5 Register

Offset: 0x230 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.42 USB2_CONTROLLER_2_USB2D_ENDPTCTRL6_0

USB2D Endpoint Control 6 Register

Offset: 0x234 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.43 USB2_CONTROLLER_2_USB2D_ENDPTCTRL7_0

USB2D Endpoint Control 7 Register

Offset: 0x238 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.44 USB2_CONTROLLER_2_USB2D_ENDPTCTRL8_0

USB2D Endpoint Control 8 Register

Offset: 0x23c | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.45 USB2_CONTROLLER_2_USB2D_ENDPTCTRL9_0

USB2D Endpoint Control 9 Register

Offset: 0x240 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.46 USB2_CONTROLLER_2_USB2D_ENDPTCTRL10_0

USB2D Endpoint Control 10 Register

Offset: 0x244 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.47 USB2_CONTROLLER_2_USB2D_ENDPTCTRL11_0

USB2D Endpoint Control 11 Register

Offset: 0x248 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.48 USB2_CONTROLLER_2_USB2D_ENDPTCTRL12_0

USB2D Endpoint Control 12 Register

Offset: 0x24c | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.49 USB2_CONTROLLER_2_USB2D_ENDPTCTRL13_0

USB2D Endpoint Control 13 Register

Offset: 0x250 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.50 USB2_CONTROLLER_2_USB2D_ENDPTCTRL14_0

USB2D Endpoint Control 14 Register

Offset: 0x254 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.7.51 USB2_CONTROLLER_2_USB2D_ENDPTCTRL15_0

USB2D Endpoint Control 15 Register

Offset: 0x258 | Read/Write: R/W | Reset: 0bxxxxxxxx000x00x0xxxxxxxx000x00x0

Bit	R/W	Reset	Description
23	RW	0x0	TXE: TX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
22	RW	0x0	TXR: TX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the Host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
21	RW	0x0	TXI: TX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
19:18	RW	0x0	TXT: TX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
17	RO	X	TXD: This bit is fixed to 0.
16	RW	0x0	TXS: TX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL
7	RW	0x0	RXE: RX Endpoint Enable. An Endpoint should be enabled only after it has been configured. 0 = DISABLE 1 = ENABLE
6	RW	0x0	RXR: RX Data Toggle Reset: Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PIDs between the host and device. 0 = KEEP_GOING 1 = RESET_PID_SEQ
5	RW	0x0	RXI: RX Data Toggle Inhibit: This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID. 0 = DIS_PID_SEQ 1 = ENB_PID_SEQ
3:2	RW	0x0	RXT: RX Endpoint Type: 00 = Control. 01 = Isochronous. 10 = Bulk. 11 = Interrupt. 0 = CTRL 1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This bit is fixed to 0.
0	RW	0x0	RXS: RX Endpoint Stall: This bit will be set automatically upon receipt of a SETUP request if this Endpoint is not configured as a Control Endpoint. It will be cleared automatically upon receipt a SETUP request if this Endpoint is configured as a Control Endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 = EP_OK 1 = EP_STALL

23.9.8 USB3 Controller Interface Registers

These are used to generate the actual RTL registers for the USB3 controller interface.

23.9.8.1 USB3_IF_USB_SUSP_CTRL_0

USB Suspend Control Register

This register controls the suspend and resume behavior of the USB controller/PHY.

Offset: 0x400 | Read/Write: R/W | Reset: 0bxxxxx00xxxxx000001001000xx000000

Bit	R/W	Reset	Description
26	RW	0x0	FAST_WAKEUP_RESP: Enable Fast Response from UTMIP PHY for a Remote Wakeup request from device. This is used only for cases where wakeup response needs to be within 1 ms of spec. Used for Host mode ONLY. 0 = DISABLE 1 = ENABLE
25	RW	0x0	UTMIP_SUSPL1_SET: Enable SuspendL1 for UTMIP PHY. Enabling this will only cut off clocks to the UTMIP logic. The USB PLLs, PllU and UTMIP PLL will still be running. 0 = DISABLE 1 = ENABLE
19	RW	0x0	UHSIC_PHY_ENB: Enable UHSIC PHY mode. Set this to 1 if using the UHSIC PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
18:16	RW	0x0	USB_WAKEUP_DEBOUNCE_COUNT: USB PHY wakeup debounce counter. The USB will debounce any wakeup event by the number of clocks programmed in this counter. A value of 0 results in no debounce.
15	RW	0x0	ICUSB_MOD_CLK_ENB: ICUSB module clock enable. Enables transceiver clock to USB controller when in ICUSB mode. After setting ICUSB_PHY_ENB, software needs to wait until PLLU output is stable before setting ICUSB_MOD_CLK_ENB to ENABLE. This will be reset to DISABLE whenever ICUSB is in suspend. Software needs to enable it after ICUSB comes out of suspend after waiting for PLLU output to be stable again. 0 = DISABLE 1 = ENABLE
14	RW	0x1	UHSIC_RESET: Reset going to UHSIC PHY (active high). This should be set to 1 whenever programming the UHSIC config registers. It should be cleared to 0 after the programming of UHSIC config registers is done. UHSIC config registers should be programmed only once before doing any transactions on UHSIC. The UHSIC PHY registers should be programmed while the UHSIC is in reset. 0 = DISABLE 1 = ENABLE
13	RW	0x0	ICUSB_PHY_ENB: Enable ICUSB PHY mode. Setting this will enable the PLLU output clock when ICUSB is not in suspend mode. 0 = DISABLE 1 = ENABLE
12	RW	0x0	UTMIP_PHY_ENB: Enable UTMIP PHY mode. Set this to 1 if using UTMIP PHY. Otherwise set this to 0. 0 = DISABLE 1 = ENABLE
11	RW	0x1	UTMIP_RESET: Reset going to UTMIP PHY (active high). This should be set to 1 whenever programming the UTMIP config registers. It should be cleared to 0 after the programming of UTMIP config registers is done. UTMIP config registers should be programmed only once before doing any transactions on USB. The UTMIP PHY registers should be programmed while UTMIP is in reset. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
10	RW	0x0	USB_SUSP_POL: Polarity of the suspend signal going to the USB PHY. 0 = Active low (default), 1 = Active high. This should not be changed by software. 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
9	RW	0x0	USB_PHY_CLK_VALID_INT_ENB: USB PHY clock valid interrupt enable. If this bit is enabled, interrupt is generated whenever USB clocks are resumed from a suspend. 0 = DISABLE 1 = ENABLE
8	RO	0x0	USB_PHY_CLK_VALID_INT_STS: USB PHY clock valid interrupt status. This bit is set whenever the USB PHY clock is woken up from suspend. Software must write a 1 to clear this bit. 0 = UNSET 1 = SET
7	RO	X	USB_PHY_CLK_VALID: USB PHY clock valid status. This bit indicates whether the USB PHY is generating a valid clock to the USB controller. If USB PHY clock is running, this bit is set to 1, else it is set to 0. 0 = UNSET 1 = SET
6	RO	X	USB_CLKEN: USB AHB clock enable status. Indicates whether the AHB clock to the USB controller is enabled or not. If AHB clock to USB controller is enabled, this bit is set to 1, else it is set to 0. NOTE: Even when this is set to 0, all essential blocks that are required to resume USB clocks from suspend will be active and their AHB clock will not be suspended. 0 = UNSET 1 = SET
5	RW	0x0	USB_SUSP_CLR: Suspend Clear. Software must write a 1 to this bit to bring the PHY out of suspend mode. This is used when the software stops the PHY clock during suspend and then wants to initiate a resume. Software should also write 0 to clear it. NOTE: It is required that software generate a positive pulse on this bit to guarantee proper operation. 0 = UNSET 1 = SET
4	RW	0x0	USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect Enable (device mode). When enabled (1), USB device will wake up from suspend on a disconnect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
3	RW	0x0	USB_WAKE_ON_CNNT_EN_DEV: Wake on Connect Enable (device mode). When enabled (1), USB device will wake up from suspend on a connect event. This is only valid when USB controller is in device mode, it is not applicable when USB controller is in host mode. 0 = DISABLE 1 = ENABLE
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable. If this bit is enabled, USB will wake up from suspend whenever a resume event is detected on USB. This is valid for both USB device and USB host modes. 0 = DISABLE 1 = ENABLE
1	RW	0x0	USB_WAKEUP_INT_ENB: USB wakeup interrupt enable. If this bit is enabled, interrupt is generated whenever USB wakeup event is generated. 0 = DISABLE 1 = ENABLE
0	RO	0x0	USB_WAKEUP_INT_STS: USB wakeup interrupt status. This bit is set whenever USB wakes up from suspend (a wakeup event is generated). Software must write a 1 to clear this bit. Note that during the wakeup sequence, PHY clocks will be resumed from suspend. Software can check when the PHY clocks are resumed by reading the bit USB_PHY_CLK_VALID. There is also a separate interrupt generated when PHY clock is resumed if USB_PHY_CLK_VALID_INT_EN is set. During the wakeup sequence, first USB_WAKEUP_INT_STS will be set, and it will take some time for the PHY clock to resume, which can be detected by checking USB_PHY_CLK_VALID. 0 = UNSET 1 = SET

23.9.8.2 USB3_IF_USB_PHY_VBUS_SENSORS_0

USB PHY VBUS SENSORS Control Register

This register controls the OTG VBUS sensors in the USB PHY. There are 4 VBUS sensors:

- A_VBUS_VLD
- A_SESS_VLD
- B_SESS_VLD
- B_SESS_END

The debounced status of each sensor can be read from the corresponding _STS bit field of the sensor in this register. The _CHG_DET field is set to 1 whenever a change is detected in the value of the _STS bit field of the corresponding sensor. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/COP by appropriately writing the USBD bits in the interrupt controller registers.

In case software wants to override the value for a sensor, it can set the corresponding _SW_EN to 1, and set the corresponding sensor _SW_VALUE to 1 or 0 as per the requirement.

There are two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

Offset: 0x404 | Read/Write: R/W | Reset: 0bx0000x00x0000x00x0000x00x0000x00

Bit	R/W	Reset	Description
30	RW	0x0	A_VBUS_VLD_WAKEUP_EN: A_VBUS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on A_VBUS_VLD. 0 = DISABLE 1 = ENABLE
29	RW	0x0	A_VBUS_VLD_DEB_SEL_B: A_VBUS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	A_VBUS_VLD_SW_VALUE: A_VBUS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_VBUS_VLD status. This is only valid when A_VBUS_VLD_SW_EN is set. 0 = UNSET 1 = SET
27	RW	0x0	A_VBUS_VLD_SW_EN: A_VBUS_VLD software enable. Enable Software Controlled A_VBUS_VLD. Software sets this bit to drive the value in A_VBUS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
26	RO	X	A_VBUS_VLD_STS: A_VBUS_VLD status. This is set to 1 whenever A_VBUS_VLD sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	A_VBUS_VLD_CHG_DET: A_VBUS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_VBUS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
24	RW	0x0	A_VBUS_VLD_INT_EN: A_VBUS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever A_VBUS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
22	RW	0x0	A_SESS_VLD_WAKEUP_EN: A_SESS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on A_SESS_VLD. 0 = DISABLE 1 = ENABLE
21	RW	0x0	A_SESS_VLD_DEB_SEL_B: A_SESS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
20	RW	0x0	A_SESS_VLD_SW_VALUE: A_SESS_VLD software value. Software should write the appropriate value (1/0) to set/unset the A_SESS_VLD status. This is only valid when A_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	A_SESS_VLD_SW_EN: A_SESS_VLD software enable. Enable Software Controlled A_SESS_VLD. Software sets this bit to drive the value in A_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
18	RO	X	A_SESS_VLD_STS: A_SESS_VLD status. This is set to 1 whenever A_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	A_SESS_VLD_CHG_DET: A_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of A_SESS_VLD. Software writes a 1 to clear it. 0 = UNSET 1 = SET
16	RW	0x0	A_SESS_VLD_INT_EN: A_SESS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever A_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
14	RW	0x0	B_SESS_VLD_WAKEUP_EN: B_SESS_VLD wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on B_SESS_VLD. 0 = DISABLE 1 = ENABLE
13	RW	0x0	B_SESS_VLD_DEB_SEL_B: B_SESS_VLD debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	B_SESS_VLD_SW_VALUE: B_SESS_VLD software value. Software should write the appropriate value (1/0) to set/unset the B_SESS_VLD status. This is only valid when B_SESS_VLD_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	B_SESS_VLD_SW_EN: B_SESS_VLD software enable. Enable Software Controlled B_SESS_VLD. Software sets this bit to drive the value in B_SESS_VLD_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	B_SESS_VLD_STS: B_SESS_VLD status. This is set to 1 whenever B_SESS_VLD sensor output is 1. 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
9	RO	0x0	B_SESS_VLD_CHG_DET: B_SESS_VLD change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_VLD. software writes a 1 to clear it 0 = UNSET 1 = SET
8	RW	0x0	B_SESS_VLD_INT_EN: B_SESS_VLD interrupt enable. If this field is set to 1, an interrupt is generated whenever B_SESS_VLD_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
6	RW	0x0	B_SESS_END_WAKEUP_EN: B_SESS_END wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on B_SESS_END. 0 = DISABLE 1 = ENABLE
5	RW	0x0	B_SESS_END_DEB_SEL_B: B_SESS_END debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
4	RW	0x0	B_SESS_END_SW_VALUE: B_SESS_END software value. Software should write the appropriate value (1/0) to set/unset the B_SESS_END status. This is only valid when B_SESS_END_SW_EN is set. 0 = UNSET 1 = SET
3	RW	0x0	B_SESS_END_SW_EN: Enable Software Controlled B_SESS_END. Software sets this bit to drive the value in B_SESS_END_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
2	RO	X	B_SESS_END_STS: B_SESS_END status. This is set to 1 whenever the B_SESS_END sensor output is 1. 0 = UNSET 1 = SET
1	RO	0x0	B_SESS_END_CHG_DET: B_SESS_END change detect. This field is set by hardware whenever a change is detected in the value of B_SESS_END. software writes a 1 to clear it 0 = UNSET 1 = SET
0	RW	0x0	B_SESS_END_INT_EN: B_SESS_END interrupt enable. If this field is set to 1, an interrupt is generated whenever B_SESS_END_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE

23.9.8.3 USB3_IF_USB_PHY_VBUS_WAKEUP_ID_0

USB PHY VBUS Wakeup and ID Control Register

This register controls the battery charger (VDCD_DET, VDAT_DET), VBUS_WAKEUP, and ID sensors. The following sensors are in this register:

- VBUS_WAKEUP
- ID
- VDAT_DET
- VDCD_DET

The debounced status of each sensor can be read from the corresponding _STS bit field of the sensor in this register. The _CHG_DET field is set to 1 whenever a change is detected in the value of the _STS bit field of the corresponding sensor. If _INT_EN is set, then an interrupt is generated to the processor. This interrupt can be routed to CPU/COP by appropriately writing the USBD bits in the interrupt controller registers.

In case software wants to override the value for a sensor, it can set the corresponding _SW_EN to 1, and set the corresponding sensor _SW_VALUE to 1 or 0 as per the requirement.

There are two debouncers for each sensor - DEBOUNCE_A and DEBOUNCE_B. The debounce values for them are controlled by the register UTMIP_DEBOUNCE_CFG0, fields UTMIP_BIAS_DEBOUNCE_A and UTMIP_BIAS_DEBOUNCE_B. For each sensor, we can select whether to use DEBOUNCE_A or DEBOUNCE_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

There are debouncers for VDAT_DET and VDCD_DET. These use separate debouncers - CHRG_DEBOUNCE_PERIOD_A and CHRG_DEBOUNCE_PERIOD_B. The debounce values for them are controlled by the register UTMIP_CHRG_DEB_CFG0, fields UTMIP_CHRG_DEBOUNCE_PERIOD_A and UTMIP_CHRG_DEBOUNCE_PERIOD_B. For each sensor, we can select whether to use HRG_DEBOUNCE_PERIOD_A or CHRG_DEBOUNCE_PERIOD_B by setting the field _DEB_SEL_B to the appropriate value (SEL_A or SEL_B).

Note: Do not set either UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B to 0x0. If not using one of the debouncers, keep it at the default value of 0xFFFF.

Offset: 0x408 | Read/Write: R/W | Reset: 0bx0000x00xx000x00xx000x00x1000x00

Bit	R/W	Reset	Description
30	RW	0x0	VBUS_WAKEUP_WAKEUP_EN: VBUS_WAKEUP wakeup enable. If this bit is enabled, USB will wake up from suspend whenever a change is detected on VBUS_WAKEUP. 0 = DISABLE 1 = ENABLE
29	RW	0x0	VDCD_DET_DEB_SEL_B: VCDT_DET debounce A/B select. Selects the debounce value from UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B from the register UTMIP_CHRG_DEB_CFG0. 0 = SEL_A 1 = SEL_B
28	RW	0x0	VDCD_DET_SW_VALUE: VDCD_DET software value. Software should write the appropriate value (1/0) to set/unset the VDCD_DET status. This is only valid when VDCD_DET_SW_EN is set. 0 = UNSET 1 = SET
27	RW	0x0	VDCD_DET_SW_EN: VDCD_DET software enable. Enable Software Controlled VDCD_DET. Software sets this bit to drive the value in VDCD_DET_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
26	RO	X	VDCD_DET_STS: VDCD_DET status. This is set to 1 whenever VDCD_DET sensor output is 1. 0 = UNSET 1 = SET
25	RO	0x0	VDCD_DET_CHG_DET: VDCD_DET change detect. This field is set by hardware whenever a change is detected in the value of VDCD_DET. Software writes a 1 to clear it. 0 = UNSET 1 = SET
24	RW	0x0	VDCD_DET_INT_EN: VDCD_DET interrupt enable. If this field is set to 1, an interrupt is generated whenever VDCD_DET_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
21	RW	0x0	VDAT_DET_DEB_SEL_B: VDAT_DET debounce A/B select. Selects between the two debounce values UTMIP_CHRG_DEBOUNCE_PERIOD_A or UTMIP_CHRG_DEBOUNCE_PERIOD_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B

Bit	R/W	Reset	Description
20	RW	0x0	VDAT_DET_SW_VALUE: VDAT_DET software value. Software should write the appropriate value (1/0) to set/unset the VDAT_DET status. This is only valid when VDAT_DET_SW_EN is set. 0 = UNSET 1 = SET
19	RW	0x0	VDAT_DET_SW_EN: Enable Software Controlled VDAT_DET. Software sets this bit to drive the value in VDAT_DET_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
18	RO	X	VDAT_DET_STS: VDAT_DET status. This is set to 1 whenever VDAT_DET sensor output is 1. 0 = UNSET 1 = SET
17	RO	0x0	VDAT_DET_CHG_DET: VDAT_DET change detect. This field is set by hardware whenever a change is detected in the value of VDAT_DET. software writes a 1 to clear it 0 = UNSET 1 = SET
16	RW	0x0	VDAT_DET_INT_EN: VDAT_DET interrupt enable. If this field is set to 1, an interrupt is generated whenever VDAT_DET_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
13	RW	0x0	VBUS_WAKEUP_DEB_SEL_B: VBUS_WAKEUP debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B
12	RW	0x0	VBUS_WAKEUP_SW_VALUE: VBUS wakeup software value. Software should write the appropriate value (1/0) to set/unset the VBUS_WAKEUP status. This is only valid when VBUS_WAKEUP_SW_EN is set. 0 = UNSET 1 = SET
11	RW	0x0	VBUS_WAKEUP_SW_EN: Enable Software Controlled VBUS_WAKEUP. Software sets this bit to drive the value in VBUS_WAKEUP_SW_VALUE to the USB controller. 0 = DISABLE 1 = ENABLE
10	RO	X	VBUS_WAKEUP_STS: VBUS wakeup status. This is set to 1 whenever VBUS_WAKEUP sensor output is 1. 0 = UNSET 1 = SET
9	RO	0x0	VBUS_WAKEUP_CHG_DET: VBUS wakeup change detect. This field is set by hardware whenever a change is detected in the value of VBUS_WAKEUP. software writes a 1 to clear it. 0 = UNSET 1 = SET
8	RW	0x0	VBUS_WAKEUP_INT_EN: VBUS wakeup interrupt enable. If this field is set to 1, an interrupt is generated whenever VBUS_WAKEUP_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE
7	RO	X	STATIC_GPI: Static GPI status 0 = UNSET 1 = SET
6	RW	0x1	ID_PU: ID pullup enable. Set to 1. 0 = DISABLE 1 = ENABLE
5	RW	0x0	ID_DEB_SEL_B: ID debounce A/B select. Selects between the two debounce values UTMIP_BIAS_DEBOUNCE_A or UTMIP_BIAS_DEBOUNCE_B from the register UTMIP_DEBOUNCE_CFG0. 0 = SEL_A 1 = SEL_B

Bit	R/W	Reset	Description
4	RW	0x0	ID_SW_VALUE: ID software value. Software should write the appropriate value (1/0) to set/unset the ID status. This is only valid when ID_SW_EN is set. 0 = UNSET 1 = SET
3	RW	0x0	ID_SW_EN: Enable Software Controlled ID. Software sets this bit to drive the value in ID_SW_VALUE to the USB controller 0 = DISABLE 1 = ENABLE
2	RO	X	ID_STS: ID status. This is set to 1 whenever ID sensor output is 1. 0 = UNSET 1 = SET
1	RO	0x0	ID_CHG_DET: ID change detect. This field is set by hardware whenever a change is detected in the value of ID. software writes a 1 to clear it 0 = UNSET 1 = SET
0	RW	0x0	ID_INT_EN: ID interrupt enable. If this field is set to 1, an interrupt is generated whenever ID_CHG_DET is set to 1. 0 = DISABLE 1 = ENABLE

23.9.8.4 USB3_IF_USB_PHY_ALT_VBUS_STS_0

USB PHY Alternate VBUS Status Register

Offset: 0x40c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
8	X	VDCD_DET_ALT: VDCD_DET alternate status 0 = UNSET 1 = SET
7	X	VDAT_DET_ALT: VDAT_DET alternate status 0 = UNSET 1 = SET
6	X	A_SESS_VLD_ALT: A_SESS_VLD alternate status 0 = UNSET 1 = SET
5	X	B_SESS_VLD_ALT: B_SESS_VLD alternate status 0 = UNSET 1 = SET
4	X	ID_DIG_ALT: ID alternate status 0 = UNSET 1 = SET
3	X	B_SESS_END_ALT: B_SESS_END alternate status 0 = UNSET 1 = SET
2	X	STATIC_GPI_ALT: Static GPI alternate status 0 = UNSET 1 = SET
1	X	A_VBUS_VLD_ALT: A_VBUS_VLD alternate status 0 = UNSET 1 = SET
0	X	VBUS_WAKEUP_ALT: Vbus wakeup alternate status 0 = UNSET 1 = SET

23.9.8.5 USB3_IF_ICUSB_XCVR_CFG_0

ICUSB Transceiver Configuration Register

Offset: 0x414 | Read/Write: R/W | Reset: 0bxxxxxxxxxxx10000xx0000000xxx1111

Bit	R/W	Reset	Description
31:24	RO	X	ICUSB_CALOUT: ICUSB PHY calibration code
20	RW	0x1	ICUSB_CALOUT_EN: ICUSB PHY auto-calibration enable 0 = DISABLE 1 = ENABLE
19:16	RW	0x0	ICUSB_DRV: ICUSB PHY Drive strength offset
13:8	RW	0x0	ICUSB_SLEW: ICUSB PHY FS/LS slew rate control
7	RW	0x0	ICUSB_SEL_DIFF_RCVR: ICUSB differential receiver select 0 = SINGLE 1 = DIFF
3	RW	0x1	ICUSB_IDDQ: ICUSB PHY IDDQ shutdown mode 0 = NORMAL 1 = OFF
2	RW	0x1	ICUSB_PD_ZI: ICUSB PHY Single-ended receiver power down 0 = NORMAL 1 = OFF
1	RW	0x1	ICUSB_PD_DR: ICUSB PHY Differential receiver power down 0 = NORMAL 1 = OFF
0	RW	0x1	ICUSB_PD_TX: ICUSB PHY Low/full-speed driver power down 0 = NORMAL 1 = OFF

23.9.8.6 USB3_IF_USB_INTER_PKT_DELAY_CTRL_0

Inter Packet Delay Control

This controls the interpacket delay between two consecutive transmit packets in HS mode. This only applies to host mode because device never transmits two packets in a row.

Offset: 0x420 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0010010

Bit	Reset	Description
6:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. This is valid only for the UTMIP PHY. Software should not change this.

23.9.8.7 USB3_IF_USB_RSM_DLY_0

USB Controller Resume Signalling Delay

Offset: 0x490 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0110100101111000

Bit	Reset	Description
15:0	0x6978	TIME_TO_RESUME: Send the resume back in number of 60 MHz cycles. Default gives 900 µs delay. Only applicable in host mode.

23.9.8.8 USB3_IF_ICUSB_PADCTLS_0

ICUSB Pad Controls Config

Offset: 0x494 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx000000000

Bit	Reset	Description
8	0x0	ICUSB_RPU2_EN: Config RPU2_EN state. 0 = DISABLE 1 = ENABLE
7	0x0	ICUSB_DISABLE_PULLUP_DP: Force DP pull-up inactive. (Overrides FORCE_PULLUP_DP.)
6	0x0	ICUSB_DISABLE_PULLUP_DM: Force DM pull-up inactive. (Overrides FORCE_PULLUP_DM.)
5	0x0	ICUSB_DISABLE_PULLDN_DP: Force DP pull-down inactive. (Overrides FORCE_PULLDN_DP.)
4	0x0	ICUSB_DISABLE_PULLDN_DM: Force DM pull-down inactive. (Overrides FORCE_PULLDN_DM.)
3	0x0	ICUSB_FORCE_PULLUP_DP: Force DP pull-up active.
2	0x0	ICUSB_FORCE_PULLUP_DM: Force DM pull-up active.
1	0x0	ICUSB_FORCE_PULLDN_DP: Force DP pull-down active.
0	0x0	ICUSB_FORCE_PULLDN_DM: Force DM pull-down active.

23.9.8.9 USB3_IF_SPARE_0

Spare Register

Offset: 0x498 | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:16	0xffff	SPARE_HI: Spare register bits, defaulted to 1. SPARE_HI[0] - usb_port_reset_fix_en. Enable port-reset fix SPARE_HI[1] - hsic_conn_det_feature_enable. enable HSIC connect detection
15:0	0x0	SPARE_LO: Spare register bits, defaulted to 0. SPARE_LO[4] - usb_hs_rsm_eop_fix_en. Enable HS_RESUME EOP fix SPARE_LO[5] - usb_port_suspend_fix_en

23.9.8.10 USB3_IF_USB1_NEW_CONTROL_0

USB Coherency and Memory Alignment Controls

For ULPI_DIR_OVERRIDE

Offset: 0x4c0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxx00001111xxxxxx00

Bit	Reset	Description
15:8	0xf	REQUEST_EXPIRY_COUNTER: Time to wait for coalescing the request.
1	0x0	MEM_ALIGNMENT_MUX_EN: Mux to select between Tegra 3 and Tegra 4 code 0 = DISABLE 1 = ENABLE
0	0x0	COHERENCY_EN: Enable fence mechanism. 0 = DISABLE 1 = ENABLE

23.9.9 XUSB Host Registers

23.9.9.1 XUSB_HOST_AXI_BAR0_SZ_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000008 (0bxxxxxxxxxxxx00000000000000001000)

Bit	Reset	Description
19:0	0x8	AXI_BAR0_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

23.9.9.2 XUSB_HOST_AXI_BAR1_SZ_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR1_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

23.9.9.3 XUSB_HOST_AXI_BAR2_SZ_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR2_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

23.9.9.4 XUSB_HOST_AXI_BAR3_SZ_0

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	AXI_BAR3_SIZE: The size of the address range associated with BARi is in 4K increments. Value of 0 signifies BARi is not used.

23.9.9.5 XUSB_HOST_AXI_BAR0_START_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x70090000 (0b01110000000010010000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x70090	AXI_BAR0_START: The start of the AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

23.9.9.6 XUSB_HOST_AXI_BAR1_START_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR1_START: The start of AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

23.9.9.7 XUSB_HOST_AXI_BAR2_START_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR2_START: The start of the AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

23.9.9.8 XUSB_HOST_AXI_BAR3_START_0

Offset: 0x4c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	AXI_BAR3_START: The start of the AXI address space for BARi. The AXI target address is compared to start/size for each BAR to determine if the access is to that BAR.

23.9.9.9 XUSB_HOST_FPCI_BAR0_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x00700901 (0b00000000011100000000010010000xxx1)

Bit	Reset	Description
31:4	0x70090	FPCI_BAR0_START: The start of the FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a 12-bit left shift of the value of this register.
0	0x1	FPCI_BAR0_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 0 = memory-mapped access (PW only) 1 = I/O or config access (NPW only)

23.9.9.10 XUSB_HOST_FPCI_BAR1_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR1_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a 12-bit left shift of the value of this register.
0	0x1	FPCI_BAR1_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 0 = Memory-mapped access (PW only) 1 = I/O or config access (NPW only)

23.9.9.11 XUSB_HOST_FPCI_BAR2_0

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR2_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a 12-bit left shift of the value of this register.
0	0x1	FPCI_BAR2_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 0 = Memory-mapped access (PW only) 1 = I/O or config access (NPW only)

23.9.9.12 XUSB_HOST_FPCI_BAR3_0

Offset: 0x8c | Read/Write: R/W | Reset: 0x00000001 (0b00000000000000000000000000000xxx1)

Bit	Reset	Description
31:4	0x0	FPCI_BAR3_START: The start of FPCI address space mapped into the BARi range of PCI memory space. The 40-bit FPCI address is determined by a 12-bit left shift of the value of this register.
0	0x1	FPCI_BAR3_ACCESS_TYPE: Indicates if the address region is memory mapped versus configuration or I/O space. 0 = Memory-mapped access (PW only) 1 = I/O or config access (NPW only)

23.9.9.13 XUSB_HOST_MSI_BAR_SZ_0

MSI BAR SIZE

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00000000000000000000)

Bit	Reset	Description
19:0	0x0	MSI_BAR_SIZE: The size of the address range associated with MSI BAR is in 4K increments. Value of 0 signifies BAR is not used.

23.9.9.14 XUSB_HOST_MSI_AXI_BAR_ST_0

MSI AXI BAR START

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000xxxxxxxxxxxx)

Bit	Reset	Description
31:12	0x0	MSI_AXI_BAR_START: The start of the upstream AXI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for the MSI BAR to determine if the access is MSI. Bits 31:12 of MSI BAR start correspond to AXI address bits 31:12.

23.9.9.15 XUSB_HOST_MSI_FPCI_BAR_ST_0

MSI FPCI BAR START

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000xxxx)

Bit	Reset	Description
31:4	0x0	MSI_FPCI_BAR_START: The start of the upstream FPCI address space for MSI BAR. The upstream FPCI address is compared to start/1KB range for MSI BAR to determine if the access is MSI. Bits 31:4 of MSI BAR start correspond to UFPCI address bits 39:12.

23.9.9.16 XUSB_HOST_MSI_VEC0_0

MSI VECTORi, i in [0,7]

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR0: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

23.9.9.17 XUSB_HOST_MSI_VEC1_0

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR1: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

23.9.9.18 XUSB_HOST_MSI_VEC2_0

Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR2: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

23.9.9.19 XUSB_HOST_MSI_VEC3_0

Offset: 0x10c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR3: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

23.9.9.20 XUSB_HOST_MSI_VEC4_0

Offset: 0x110 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR4: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

23.9.9.21 XUSB_HOST_MSI_VEC5_0

Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR5: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

23.9.9.22 XUSB_HOST_MSI_VEC6_0

Offset: 0x118 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR6: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

23.9.9.23 XUSB_HOST_MSI_VEC7_0

Offset: 0x11c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_VECTOR7: Each vector register corresponds to 32 of the possible 256 MSI vectors.VECTOR0 corresponds to MSI vectors 31-0. Vector7 corresponds to MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.The bit is set to 0 if a 1 is written to its location.

23.9.9.24 XUSB_HOST_MSI_EN_VEC0_0

MSI ENABLE VECTORi, i in [0,7], RW

Offset: 0x140 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR0: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

23.9.9.25 XUSB_HOST_MSI_EN_VEC1_0

Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR1: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

23.9.9.26 XUSB_HOST_MSI_EN_VEC2_0

Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR2: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

23.9.9.27 XUSB_HOST_MSI_EN_VEC3_0

Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR3: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

23.9.9.28 XUSB_HOST_MSI_EN_VEC4_0

Offset: 0x150 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR4: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

23.9.9.29 XUSB_HOST_MSI_EN_VEC5_0

Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR5: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

23.9.9.30 XUSB_HOST_MSI_EN_VEC6_0

Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR6: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

23.9.9.31 XUSB_HOST_MSI_EN_VEC7_0

Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	MSI_ENABLE_VECTOR7: Each vector register corresponds to the enable bit for 32 of the possible 256 MSI vectors. ENABLE VECTOR0 corresponds to enable bits for MSI vectors 31-0. Vector7 corresponds to enable bits for MSI vectors 255-223.When an upstream MSI is sent, the bit corresponding to the MSI vector is set to 1 by hardware if the corresponding enable bit is 1.

23.9.9.32 XUSB_HOST_CONFIGURATION_0

Configuration

Offset: 0x180 | Read/Write: R/W | Reset: 0x800X8X00 (0b1xxxxxxxxxxxxxxx10xxxxxx00000000)

Bit	R/W	Reset	Description
31	RW	0x1	CLKEN_OVERRIDE: This can override the clock enable in case of a malfunction.
18	RO	X	INITIATOR_READ_IDLE: This read-only bit provides status reads on AFI upstream A value of 1b indicates there are no outstanding reads to the initiator.
17	RO	X	INITIATOR_WRITE_IDLE: This read-only bit provides status writes on AFI upstream A value of 1b indicates there are no outstanding writes to the initiator.
15	RW	0x1	WDATA_LEAD_CYA: Used to en(dis)able the handling of write data ahead of requests on IPFS AXI target.
14	RW	0x0	WR_INTRLV_CYA: Used to en(dis)able the handling of interleaved write requests on IPFS AXI target.

Bit	R/W	Reset	Description
11	RO	X	TARGET_READ_IDLE: This read-only bit provides status reads to IPFS target. A value of 1b indicates there are no outstanding reads to the downstream FPCI.
10	RO	X	TARGET_WRITE_IDLE: This read-only bit provides status writes to IPFS target. A value of 1b indicates there are no outstanding writes to the downstream FPCI.
9	RO	X	MSI_VEC_EMPTY: This read-only bit provides status on whether MSI Vector registers have any active bits valid or not.
7	RW	0x0	UFPCI_DUAL_CHANNEL_BLOCK: Input to the upstream requests 1 = Enable dual-channel blocking 0 = Disable dual-channel blocking
6	RW	0x0	UFPCI_PWPASSPW: Input to the upstream FPCI 1 = Whenever write is ready, send it. 0 = Write goes only when outstanding PWs outside of new write's region are retired (default).
5	RW	0x0	UFPCI_PASSPW: Input to the upstream FPCI. Allows the upstream FPCI reads to pass writes.
4	RW	0x0	UFPCI_PWPASSNPW: Used for the upstream FPCI. Allows the upstream FPCI PWs to pass NPWs.
3	RW	0x0	DFPCI_PWPASSNPW: Used for the downstream FPCI. Allows the downstream FPCI PWs to pass NPWs.
2	RW	0x0	DFPCI_RSPPASSPW: Input to the downstream FPCI. Allows the downstream FPCI responses to pass writes
1	RW	0x0	DFPCI_PASSPW: Input to the downstream FPCI. Allow the downstream FPCI reads to pass writes.
0	RW	0x0	EN_FPCI: When the IPFS device block is disabled, it is completely invisible on the IPFS bus; i.e., it does not even process IPFS configuration accesses.

23.9.9.33 XUSB_HOST_FPCI_ERROR_MASKS_0

FPCI Error Masks

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	Reset	Description
2	0x0	MASK_FPCI_MASTER_ABORT: This bit allows an FPCI error to be forwarded to an AXI response when the FPCI error response indicates a Master Abort. 1 = Forward error 0 = Return AXI OKAY response (2'b0)
1	0x0	MASK_FPCI_DATA_ERROR: This bit allows an FPCI error to be forwarded to an AXI response when the FPCI error response indicates a Data Error. 1 = Forward error 0 = Return AXI OKAY response (2'b0)
0	0x0	MASK_FPCI_TARGET_ABORT: This bit allows an FPCI error to be forwarded to an AXI response when FPCI error response indicates a Target Abort. This bit also covers a decode error generated when there is no devsel received. 1 = Forward error 0 = Return AXI OKAY response (2'b0)

23.9.9.34 XUSB_HOST_INTR_MASK_0

Interrupt Masks

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxx0xxxxxxx0)

Bit	Reset	Description
16	0x0	IP_INT_MASK: IP (SATA/AZA) interrupt to the Cortex™-A15 MPCore™ gated by the mask.
8	0x0	MSI_MASK: MSI to the Cortex-A15 MPCore gated by the mask.
0	0x0	INT_MASK: Interrupt to the Cortex-A15 MPCore gated by the mask.

23.9.9.35 XUSB_HOST_INTR_CODE_0

Interrupt Control

Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:0	0x0	INT_CODE: Eight interrupt codes. If the code is 0, logging of the next interrupt is enabled 0 = INT_CODE_CLEAR : Clear interrupt code 1 = INT_CODE_INI_SLVERR: Interrupt code for MPCore AXI SLVERR response to IPFS 2 = INT_CODE_INI_DECERR: Interrupt code for MPCore AXI DECERR response to IPFS 3 = INT_CODE_TGT_SLVERR: Interrupt code for PCIe endpoint FPCI target abort or data error response to IPFS 4 = INT_CODE_TGT_DECERR: Interrupt code for PCIe2 FPCI master abort response to IPFS 5 = INT_CODE_TGT_WRERR: Interrupt code for bufferable write to non-posted write address region 6 = RSVD1: Reserved 7 = INT_CODE_DFPCI_DECERR: Interrupt code for PCIe2 response to downstream request when downstream FPCI address does not fall in a claimable downstream region 8 = INT_CODE_AXI_DECERR: Interrupt code for IPFS response to downstream request when AXI target AXI address does not fall in any IPFS downstream BARs 9 = INT_CODE_FPCI_TIMEOUT: Interrupt code for FPCI Timeout 10 = RSVD2: Reserved for future expansion 11 = RSVD3 12 = RSVD4 13 = RSVD5 14 = RSVD6 15 = INT_CODE_SM_FATAL_ERROR: Interrupt code for SM fatal error 16 = INT_CODE_SM_NON_FATAL_ERROR: Interrupt code for SM non-fatal error

23.9.9.36 XUSB_HOST_INTR_SIGNATURE_0

Interrupt Signature

Offset: 0x190 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000x0)

Bit	Reset	Description
31:2	0x0	INT_INFO: For interrupt codes 1-5/7-8, this field contains address bits [31:2], either in FPCI memory space or AXI space. For FPCI generated errors, the field contains the FPCI address. For AXI/IPFS generated errors, the field contains the AXI address.
0	0x0	DIR: Indicates the direction of the AXI/FPCI transaction. 1=RD/0=WRIF signature type is 6 (sideband message), this field is 1. 0 = WRITE: Interrupt due to a write transaction 1 = READ: Interrupt due to a read transaction

23.9.9.37 XUSB_HOST_UPPER_FPCI_ADDR_0

Upper FPCI Address

Offset: 0x194 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	INT_INFO_UPPER: These 8 bits are the upper byte of the captured FPCI address (bits [39:32]) when the interrupt code is 3, 4, or 7. These bits determine the region in the Hypertransport Address Map that was accessed.

23.9.9.38 XUSB_HOST_IPFS_INTR_ENABLE_0

IPFS Interrupt Enable

Offset: 0x198 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00xxxx00000000)

Bit	Reset	Description
13	0x0	EN_SM_NON_FATAL_ERROR: Enable bit for interrupt code 15
12	0x0	EN_SM_FATAL_ERROR: Enable bit for interrupt code 14
7	0x0	EN_FPCI_TIMEOUT: Enable bit for interrupt code 9
6	0x0	EN_AXI_DECERR: Enable bit for interrupt code 8
5	0x0	EN_DFPCI_DECERR: Enable bit for interrupt code 7
4	0x0	EN_TGT_WRERR: Enable bit for interrupt code 5
3	0x0	EN_TGT_DECERR: Enable bit for interrupt code 4
2	0x0	EN_TGT_SLVERR: Enable bit for interrupt code 3
1	0x0	EN_INI_DECERR: Enable bit for interrupt code 2
0	0x0	EN_INI_SLVERR: Enable bit for interrupt code 1

23.9.9.39 XUSB_HOST_UFPCI_CONFIG_0

Upstream FPCI Configuration

Offset: 0x19c | Read/Write: R/W | Reset: 0x0000000a (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx01010)

Bit	Reset	Description
4:0	0xa	UNITID_T0C0: Upstream FPCI Unit ID for controller 0. HyperTransport, upstream FPCI request

23.9.9.40 XUSB_HOST_CFG_REVID_0

CFG_REVID Register

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x000X10XX (0bxxxxxxxxxxxxxxxxxx0100xxxxxxx1xx)

Bit	R/W	Reset	Description
19	RO	X	DEV2SM_NONISO_REQUEST_PEND: This bit indicates if there is a non-ISO request pending. 0 = NO 1 = YES
18	RO	X	DEV2SM_ISO_REQUEST_PEND: This bit indicates if there is an ISO request pending. 0 = NO 1 = YES

Bit	R/W	Reset	Description
13:12	RW	0x1	STRAP_CPU_MODE: MCP: Mode to send MSI. It can be programmable. 0 = NB_INTEL 1 = NB_AMD 2 = AMD 3 = TMTA
11	RW	0x0	CFG_REVID_WRITE_ENABLE: MCP: The enable to override the rev ID. It can be programmable. 0 = CLEAR 1 = SET
10	RW	0x0	CFG_REVID_OVERRIDE: MCP: Provides a way to override the current revision ID. It can be programmable. 0 = DISABLE 1 = ENABLE
4	RO	X	DEV2LEG_NONCOH_REQUEST_PEND: MCP: Tells the leg block that a non-coherent request is pending. 0 = NO 1 = YES
3	RO	X	DEV2LEG_COH_REQUEST_PEND: MCP comment: Tells the leg block that a coherent request is pending. 0 = NO 1 = YES
2	RW	0x1	SM2DEV_FPCI_TIMEOUT_EN: FPCI timeout enable bit for Controller 0 = DISABLE 1 = ENABLE

23.9.9.41 XUSB_HOST_FPCI_TIMEOUT_0

FPCI_TIMEOUT Register

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x000f0000 (0bxxxxxxxxxxxxx11110000000000000000)

Bit	Reset	Description
19:0	0xf0000	SM2ALL_FPCI_TIMEOUT_THRESH: This field sets the timeout threshold value for the FPCI bus. It starts counting for each queue (ISO/NISO- RD/WR) with a pending request in the FPCI wrapper, the count resets when the requests are popped.

23.9.9.42 XUSB_HOST_TOM_0

Top of Memory Limit

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x3fff0fff (0bxx11111111111111xxxx111111111111)

Bit	Reset	Description
29:16	0x3fff	LEG2ALL_TOM2: Top of Memory Limit 2.
11:0	0xfff	LEG2ALL_TOM1: Top of Memory Limit 1.

23.9.9.43 XUSB_HOST_INITIATOR_ISO_PW_RESP_PENDING_0

Initiator ISO PW Response Pending

Offset: 0x1ac | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_ISO_PW_RESP_PEND: Number of pending initiator ISO PW responses

23.9.9.44 XUSB_HOST_INITIATOR_NISO_PW_RESP_PENDING_0

Initiator Non-ISO PW Response Pending

Offset: 0x1b0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	NUM_INITIATOR_NISO_PW_RESP_PEND: Number of pending initiator NISO PW responses

23.9.9.45 XUSB_HOST_INTR_STATUS_0

IPFS Interrupt Status

Offset: 0x1b4 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	IP_INTR_STATUS: Status of IP (SATA/AZA) interrupt
1	X	MSI_INTR_STATUS: Status of MSI interrupt
0	X	IPFS_INTR_STATUS: Status of IPFS interrupt

23.9.9.46 XUSB_HOST_DFPCI_BEN_0

Downstream FPCI Byte Enables

Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
31	0x0	EN_DFPCI_BEN: Enable bit for BEN. When set, the programmed BE is sent on the DFPCI bus
3:0	0x0	DFPCI_BYTE_ENABLE_N: Active low byte enables

23.9.9.47 XUSB_HOST_CLKGATE_HYSTERESIS_0

Offset: 0x1bc | Read/Write: R/W | Reset: 0x00000014 (0bxxxxxxxxxxxxxxxxxxxxxxxx00010100)

Bit	Reset	Description
7:0	0x14	CLK_DISABLE_CNT: Number of IPFS clock cycles to wait after clock gating criteria are met to disable IPFS/FPCI clocks

23.9.9.48 XUSB_HOST_XUSB_HOST_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register

Note: The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility.

The clock enable fields of this register control the second-level clock gating for the MC side of the MCCIF.

A '1' written to the override field will result in that clock reverting to the legacy mode of operation (where the msc*clk is on whenever the client clk is enabled.

Offset: 0x1dc | Byte Offset: 0x770 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx00xxxxxxxxxxx0000)

Bit	Reset	Description
17	0x0	XUSB_HOST_RCLK_OVERRIDE

Bit	Reset	Description
16	0x0	XUSB_HOST_WCLK_OVERRIDE
3	DISABLE	XUSB_HOST_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	XUSB_HOST_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	XUSB_HOST_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	XUSB_HOST_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

23.9.10 XUSB PADCTL Registers

23.9.10.1 XUSB_PADCTL_BOOT_MEDIA_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4:1	0x0	BOOT_PORT: 0 = USB2_OTG0 1 = USB2_OTG1 2 = USB2_OTG2 3 = USB2_OTG3 4 = USB2_OTG4 5 = USB2_OTG5 6 = USB2_OTG6 7 = ULPI 8 = ICUSB 9 = HSIC0 10 = HSIC1
0	0x0	BOOT_MEDIA_ENABLE: 0 = NO 1 = YES

23.9.10.2 XUSB_PADCTL_USB2_PAD_MUX_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx00000x0xxxxxxx0000)

Bit	Reset	Description
18	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK1: 0 = NOT_DISABLED 1 = DISABLED
17	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE_MASK0: 0 = NOT_DISABLED 1 = DISABLED
16	0x0	FORCE_PCIE_PAD_IDDQ_DISABLE: 0 = NOT_DISABLED 1 = DISABLED
15	0x0	USB2_HSIC_PAD_PORT1: 0 = SNPS 1 = XUSB

Bit	Reset	Description
14	0x0	USB2_HSIC_PAD_PORT0: 0 = SNPS 1 = XUSB
12	0x0	USB2_ULPI_PAD_PORT: 0 = SNPS 1 = XUSB
3:2	0x0	USB2_OTG_PAD_PORT1: 0 = SNPS 1 = XUSB 2 = UART
1:0	0x0	USB2_OTG_PAD_PORT0: 0 = SNPS 1 = XUSB 2 = UART

23.9.10.3 XUSB_PADCTL_USB2_PORT_CAP_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00xxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
25	0x0	ULPI_PORT_INTERNAL: 0 = NO 1 = YES
24	0x0	ULPI_PORT_CAP: 0 = ULPI_MASTER 1 = ULPI_PHY
7	0x0	PORT1_REVERSE_ID: 0 = NO 1 = YES
6	0x0	PORT1_INTERNAL: 0 = NO 1 = YES
5:4	0x0	PORT1_CAP: 0 = DISABLED 1 = HOST_ONLY 2 = DEVICE_ONLY 3 = OTG_CAP
3	0x0	PORT0_REVERSE_ID: 0 = NO 1 = YES
2	0x0	PORT0_INTERNAL: 0 = NO 1 = YES
1:0	0x0	PORT0_CAP: 0 = DISABLED 1 = HOST_ONLY 2 = DEVICE_ONLY 3 = OTG_CAP

23.9.10.4 XUSB_PADCTL_SNPS_OC_MAP_0

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8:6	0x0	CONTROLLER3_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 7 = OC_DETECTION_DISABLED
5:3	0x0	CONTROLLER2_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 7 = OC_DETECTION_DISABLED
2:0	0x0	CONTROLLER1_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 7 = OC_DETECTION_DISABLED

23.9.10.5 XUSB_PADCTL_USB2_OC_MAP_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5:3	0x0	PORT1_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 7 = OC_DETECTION_DISABLED
2:0	0x0	PORT0_OC_PIN: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1 7 = OC_DETECTION_DISABLED

23.9.10.6 XUSB_PADCTL_SS_PORT_MAP_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx000x000)

Bit	Reset	Description
6:4	0x0	PORT1_MAP: 1 = USB2_PORT0 2 = USB2_PORT1
2:0	0x0	PORT0_MAP: 1 = USB2_PORT0 2 = USB2_PORT1

23.9.10.7 XUSB_PADCTL_OC_DET_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxx000000xx00000000000000xxxx0000)

Bit	Reset	Description
29	0x0	OC_DETECTED_INTERRUPT_ENABLE_VBUSPAD1: 0 = NO 1 = YES
28	0x0	OC_DETECTED_INTERRUPT_ENABLE_VBUSPAD0: 0 = NO 1 = YES
27	0x0	OC_DETECTED_INTERRUPT_ENABLE3: 0 = NO 1 = YES
26	0x0	OC_DETECTED_INTERRUPT_ENABLE2: 0 = NO 1 = YES
25	0x0	OC_DETECTED_INTERRUPT_ENABLE1: 0 = NO 1 = YES
24	0x0	OC_DETECTED_INTERRUPT_ENABLE0: 0 = NO 1 = YES
21	0x0	OC_DETECTED_VBUS_PAD1: 0 = NO 1 = YES
20	0x0	OC_DETECTED_VBUS_PAD0: 0 = NO 1 = YES
19	0x0	OC_DETECTED3: 0 = NO 1 = YES
18	0x0	OC_DETECTED2: 0 = NO 1 = YES
17	0x0	OC_DETECTED1: 0 = NO 1 = YES
16	0x0	OC_DETECTED0: 0 = NO 1 = YES
15:13	0x0	VBUS_ENABLE1_OC_MAP: 0 = OC_DETECTED0 1 = OC_DETECTED1

Bit	Reset	Description
		2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1
12:10	0x0	VBUS_ENABLE0_OC_MAP: 0 = OC_DETECTED0 1 = OC_DETECTED1 2 = OC_DETECTED2 3 = OC_DETECTED3 4 = OC_DETECTED_VBUS_PAD0 5 = OC_DETECTED_VBUS_PAD1
9	0x0	VBUS_ENABLE1: 0 = NO 1 = YES
8	0x0	VBUS_ENABLE0: 0 = NO 1 = YES
3	0x0	SET_OC_DETECTED3: 0 = NO 1 = YES
2	0x0	SET_OC_DETECTED2: 0 = NO 1 = YES
1	0x0	SET_OC_DETECTED1: 0 = NO 1 = YES
0	0x0	SET_OC_DETECTED0: 0 = NO 1 = YES

23.9.10.8 XUSB_PADCTL_ELPG_PROGRAM_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00770000 (0bxxxxxxxxx111x11100x00x0000x00x00)

Bit	Reset	Description
22	0x1	SSP1_ELPG_VCORE_DOWN: 0 = NO 1 = YES
21	0x1	SSP1_ELPG_CLAMP_EN_EARLY: 0 = NO 1 = YES
20	0x1	SSP1_ELPG_CLAMP_EN: 0 = NO 1 = YES
18	0x1	SSP0_ELPG_VCORE_DOWN: 0 = NO 1 = YES
17	0x1	SSP0_ELPG_CLAMP_EN_EARLY: 0 = NO 1 = YES
16	0x1	SSP0_ELPG_CLAMP_EN: 0 = NO 1 = YES
15	0x0	SS_PORT1_WAKEUP_EVENT: 0 = NO 1 = YES

Bit	Reset	Description
14	0x0	SS_PORT0_WAKEUP_EVENT: 0 = NO 1 = YES
12	0x0	USB2_HSIC_PORT1_WAKEUP_EVENT: 0 = NO 1 = YES
11	0x0	USB2_HSIC_PORT0_WAKEUP_EVENT: 0 = NO 1 = YES
9	0x0	USB2_PORT1_WAKEUP_EVENT: 0 = NO 1 = YES
8	0x0	USB2_PORT0_WAKEUP_EVENT: 0 = NO 1 = YES
7	0x0	SS_PORT1_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
6	0x0	SS_PORT0_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
4	0x0	USB2_HSIC_PORT1_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
3	0x0	USB2_HSIC_PORT0_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
1	0x0	USB2_PORT1_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES
0	0x0	USB2_PORT0_WAKE_INTERRUPT_ENABLE: 0 = NO 1 = YES

23.9.10.9 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD0_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00XX00XX (0b000000000x000x000000000000x000x1)

Bit	R/W	Reset	Description
31	RW	0x0	GENERATE_SRP: 0 = NO 1 = YES
30	RW	0x0	SRP_INTR_EN: 0 = NO 1 = YES
29	RW	0x0	SRP_DETECTED: 0 = NO 1 = YES
28	RW	0x0	SRP_DETECT_EN: 0 = NO 1 = YES
27	RW	0x0	DCD_INTR_EN: 0 = NO 1 = YES

Bit	R/W	Reset	Description
26	RW	0x0	DCD_DETECTED: 0 = NO 1 = YES
25	RW	0x0	ZIN_FILTER_EN: 0 = NO 1 = YES
24	RW	0x0	ZIN_CHNG_INTR_EN: 0 = NO 1 = YES
23	RW	0x0	ZIN_ST_CHNG: 0 = NO 1 = YES
22	RO	X	ZIN: 0 = NO 1 = YES
21	RW	0x0	ZIP_FILTER_EN: 0 = NO 1 = YES
20	RW	0x0	ZIP_CHNG_INTR_EN: 0 = NO 1 = YES
19	RW	0x0	ZIP_ST_CHNG: 0 = NO 1 = YES
18	RO	X	ZIP: 0 = NO 1 = YES
17	RW	0x0	USBON_RPU: 0 = NO 1 = YES
16	RW	0x0	USBON_RPD: 0 = NO 1 = YES
15	RW	0x0	USBOP_RPU: 0 = NO 1 = YES
14	RW	0x0	USBOP_RPD: 0 = NO 1 = YES
13	RW	0x0	OP_I_SRC_EN: 0 = NO 1 = YES
12	RW	0x0	ON_SRC_EN: 0 = NO 1 = YES
11	RW	0x0	ON_SINK_EN: 0 = NO 1 = YES
10	RW	0x0	OP_SRC_EN: 0 = NO 1 = YES
9	RW	0x0	OP_SINK_EN: 0 = NO 1 = YES

Bit	R/W	Reset	Description
8	RW	0x0	VDAT_DET_FILTER_EN: 0 = NO 1 = YES
7	RW	0x0	VDAT_DET_CHNG_INTR_EN: 0 = NO 1 = YES
6	RW	0x0	VDAT_DET_ST_CHNG: 0 = NO 1 = YES
5	RO	X	VDAT_DET: 0 = NO 1 = YES
4	RW	0x0	VDCD_DET_FILTER_EN: 0 = NO 1 = YES
3	RW	0x0	VDCD_DET_CHNG_INTR_EN: 0 = NO 1 = YES
2	RW	0x0	VDCD_DET_ST_CHNG: 0 = NO 1 = YES
1	RO	X	VDCD_DET: 0 = NO 1 = YES
0	RW	0x1	PD_CHG: 0 = NO 1 = YES

23.9.10.10 XUSB_PADCTL_USB2_BATTERY_CHRG_OTGPAD1_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x00XX00XX (0b000000000x000x000000000000x000x1)

Bit	R/W	Reset	Description
31	RW	0x0	GENERATE_SRP: 0 = NO 1 = YES
30	RW	0x0	SRP_INTR_EN: 0 = NO 1 = YES
29	RW	0x0	SRP_DETECTED: 0 = NO 1 = YES
28	RW	0x0	SRP_DETECT_EN: 0 = NO 1 = YES
27	RW	0x0	DCD_INTR_EN: 0 = NO 1 = YES
26	RW	0x0	DCD_DETECTED: 0 = NO 1 = YES
25	RW	0x0	ZIN_FILTER_EN: 0 = NO 1 = YES

Bit	R/W	Reset	Description
24	RW	0x0	ZIN_CHNG_INTR_EN: 0 = NO 1 = YES
23	RW	0x0	ZIN_ST_CHNG: 0 = NO 1 = YES
22	RO	X	ZIN: 0 = NO 1 = YES
21	RW	0x0	ZIP_FILTER_EN: 0 = NO 1 = YES
20	RW	0x0	ZIP_CHNG_INTR_EN: 0 = NO 1 = YES
19	RW	0x0	ZIP_ST_CHNG: 0 = NO 1 = YES
18	RO	X	ZIP: 0 = NO 1 = YES
17	RW	0x0	USBON_RPU: 0 = NO 1 = YES
16	RW	0x0	USBON_RPD: 0 = NO 1 = YES
15	RW	0x0	USBOP_RPU: 0 = NO 1 = YES
14	RW	0x0	USBOP_RPD: 0 = NO 1 = YES
13	RW	0x0	OP_I_SRC_EN: 0 = NO 1 = YES
12	RW	0x0	ON_SRC_EN: 0 = NO 1 = YES
11	RW	0x0	ON_SINK_EN: 0 = NO 1 = YES
10	RW	0x0	OP_SRC_EN: 0 = NO 1 = YES
9	RW	0x0	OP_SINK_EN: 0 = NO 1 = YES
8	RW	0x0	VDAT_DET_FILTER_EN: 0 = NO 1 = YES
7	RW	0x0	VDAT_DET_CHNG_INTR_EN: 0 = NO 1 = YES

Bit	R/W	Reset	Description
6	RW	0x0	VDAT_DET_ST_CHNG: 0 = NO 1 = YES
5	RO	X	VDAT_DET: 0 = NO 1 = YES
4	RW	0x0	VDCD_DET_FILTER_EN: 0 = NO 1 = YES
3	RW	0x0	VDCD_DET_CHNG_INTR_EN: 0 = NO 1 = YES
2	RW	0x0	VDCD_DET_ST_CHNG: 0 = NO 1 = YES
1	RO	X	VDCD_DET: 0 = NO 1 = YES
0	RW	0x1	PD_CHG: 0 = NO 1 = YES

23.9.10.11 XUSB_PADCTL_USB2_BATTERY_CHRG_BIASPAD_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxx00000000xxxxxx00x00x0)

Bit	R/W	Reset	Description
20	RW	0x0	ID_OVERRIDE: 0 = NO 1 = YES
19:18	RW	0x0	ID_SOURCE_SELECT: 0 = NO_OVERRIDE 1 = GPIO 2 = VBUS_OVERRIDE
17	RW	0x0	VBUS_OVERRIDE: 0 = NO 1 = YES
16:15	RW	0x0	VBUS_SOURCE_SELECT: 0 = NO_OVERRIDE 1 = GPIO 2 = VBUS_OVERRIDE
14	RW	0x0	ID_CONNECT_CHNG_INTR_EN: 0 = NO 1 = YES
13	RW	0x0	ID_CONNECT_ST_CHNG: 0 = NO 1 = YES
12	RO	X	ID_CONNECT_STATUS: 0 = NO 1 = YES
11	RO	X	IDDIG_C: 0 = NO 1 = YES

Bit	R/W	Reset	Description
10	RO	X	IDDIG_B: 0 = NO 1 = YES
9	RO	X	IDDIG_A: 0 = NO 1 = YES
8	RO	X	IDDIG: 0 = NO 1 = YES
6	RW	0x0	VBUS_VLD_CHNG_INTR_EN: 0 = NO 1 = YES
5	RW	0x0	VBUS_VLD_ST_CHNG: 0 = NO 1 = YES
4	RO	X	VBUS_VLD: 0 = NO 1 = YES
3	RW	0x0	OTG_VBUS_SESS_VLD_CHNG_INTR_EN: 0 = NO 1 = YES
2	RW	0x0	OTG_VBUS_SESS_VLD_ST_CHNG: 0 = NO 1 = YES
1	RO	X	OTG_VBUS_SESS_VLD: 0 = NO 1 = YES
0	RW	0x0	PD_OTG: 0 = NO 1 = YES

23.9.10.12 XUSB_PADCTL_USB2_BATTERY_CHRG_TDCD_DBNC_TIMER_0

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000000)

Bit	Reset	Description
10:0	0x0	TDCD_DBNC

23.9.10.13 XUSB_PADCTL_IOPHY_PLL0_CTL1_0

USB3/PCIE PLL0 control signals: One set per PLL

Offset: 0x30 | Read/Write: R/W | Reset: 0x2X2X0000 (0bxx10xxx0xx10xxx000000x0000000000)

Bit	R/W	Reset	Description
29:28	RW	0x2	PLL1_REFCLK_NDIV
27	RO	X	PLL1_LOCKDET
24	RW	0x0	PLL1_MODE
21:20	RW	0x2	PLL0_REFCLK_NDIV
19	RO	X	PLL0_LOCKDET
16	RW	0x0	PLL0_MODE
15:12	RW	0x0	REFCLK_SEL

Bit	R/W	Reset	Description
11	RW	0x0	REFCLK_TERM100
9	RW	0x0	PLL_CKBUFPD_OVR
8	RW	0x0	PLL_CKBUFPD_M
7	RW	0x0	PLL_CKBUFPD_BL
6	RW	0x0	PLL_CKBUFPD_BR
5	RW	0x0	PLL_CKBUFPD_TL
4	RW	0x0	PLL_CKBUFPD_TR
3	RW	0x0	PLL_PWR_OVRD
2	RW	0x0	PLL_EMULATION_RST_
1	RW	0x0	PLL_RST_
0	RW	0x0	PLL_IDDQ

23.9.10.14 XUSB_PADCTL_IOPHY_PLL0_CTL2_0

Offset: 0x34 | Read/Write: R/W | Reset: 0xXX880037 (0bxxxxxxxx100010000x00000000110111)

Bit	R/W	Reset	Description
31:24	RO	X	PLL_MISC_OUT
23:20	RW	0x8	PLL1_CP_CNTL
19:16	RW	0x8	PLL0_CP_CNTL
15	RW	0x0	PLL_BYPASS_EN
13	RW	0x0	PLL_EMULATION_ON
12	RW	0x0	TCLKOUT_EN
11:8	RW	0x0	TCLKOUT_SEL
7	RW	0x0	XDIGCLK4P5_EN
6	RW	0x0	REFCLKBUF_EN
5	RW	0x1	TXCLKREF_EN
4	RW	0x1	TXCLKREF_SEL
3	RW	0x0	XDIGCLK_EN
2:0	RW	0x7	XDIGCLK_SEL

23.9.10.15 XUSB_PADCTL_IOPHY_PLL0_CTL3_0

Offset: 0x38 | Read/Write: R/W | Reset: 0x0000XX0e (0b0000xx000000xx00x0xxxxxx0xx01110)

Bit	R/W	Reset	Description
31:28	RW	0x0	PLL_TEMP_CNTL
25:20	RW	0x0	PLL_BW_CNTL
17:16	RW	0x0	PLL_BGAP_CNTL
15	RO	X	RCAL_DONE
14	RW	0x0	RCAL_RESET
12:8	RO	X	RCAL_VAL

Bit	R/W	Reset	Description
7	RW	0x0	RCAL_BYPASS
4:0	RW	0xe	RCAL_CODE

23.9.10.16 XUSB_PADCTL_IOPHY_PLL0_CTL4_0

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11:0	0x0	PLL_MISC_CNTL

23.9.10.17 XUSB_PADCTL_IOPHY_USB3_PAD0_CTL_1_0

USB3 PADS: XUSB Static controls: Unit specific

Offset: 0x40 | Read/Write: R/W | Reset: 0x0000ec14 (0bxxxxxxxxxxxxxx0001110110000010100)

Bit	Reset	Description
18:15	0x1	TX_DRV_CNTL
14:11	0xd	TX_CMADJ
10:5	0x20	TX_AMP
4:3	0x2	RX_RATE
2:1	0x2	TX_RATE
0	0x0	RATE_MODE

23.9.10.18 XUSB_PADCTL_IOPHY_USB3_PAD1_CTL_1_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x0000ec14 (0bxxxxxxxxxxxxxx0001110110000010100)

Bit	Reset	Description
18:15	0x1	TX_DRV_CNTL
14:11	0xd	TX_CMADJ
10:5	0x20	TX_AMP
4:3	0x2	RX_RATE
2:1	0x2	TX_RATE
0	0x0	RATE_MODE

23.9.10.19 XUSB_PADCTL_IOPHY_USB3_PAD0_CTL_2_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x24001000 (0b00100100000000000001000000000000)

Bit	Reset	Description
31:24	0x24	CDR_CNTL
23:8	0x10	RX_EQ
7:4	0x0	RX_WANDER
3:2	0x0	RX_TERM_CNTL
1:0	0x0	TX_TERM_CNTL

23.9.10.20 XUSB_PADCTL_IOPHY_USB3_PAD1_CTL_2_0

Offset: 0x4c | Read/Write: R/W | Reset: 0x24001000 (0b00100100000000000001000000000000)

Bit	Reset	Description
31:24	0x24	CDR_CNTL
23:8	0x10	RX_EQ
7:4	0x0	RX_WANDER
3:2	0x0	RX_TERM_CNTL
1:0	0x0	TX_TERM_CNTL

23.9.10.21 XUSB_PADCTL_IOPHY_USB3_PAD0_CTL_3_0

Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	EOM_CNTL

23.9.10.22 XUSB_PADCTL_IOPHY_USB3_PAD1_CTL_3_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	EOM_CNTL

23.9.10.23 XUSB_PADCTL_IOPHY_USB3_PAD0_CTL_4_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DFE_CNTL

23.9.10.24 XUSB_PADCTL_IOPHY_USB3_PAD1_CTL_4_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DFE_CNTL

23.9.10.25 XUSB_PADCTL_IOPHY_MISC_PAD0_CTL_1_0

USB3 PADS: Miscellaneous Dir Control: Common for all USB3/PCIE/SATA

USB3 PADS: Miscellaneous Ovrd: Common for all USB3/PCIE/SATA

Offset: 0x60 | Read/Write: R/W | Reset: 0x000aX330 (0bxxxx00000000010100xxx001100110000)

Bit	R/W	Reset	Description
27	RW	0x0	RX_PWR_OVRD
26	RW	0x0	TX_PWR_OVRD
25	RW	0x0	RATE_MODE_OVRD
24	RW	0x0	RATE_MODE

Bit	R/W	Reset	Description
23:22	RW	0x0	RX_DIV
21:20	RW	0x0	TX_DIV
19:18	RW	0x2	RX_RATE
17:16	RW	0x2	TX_RATE
15	RW	0x0	TX_RDET
13	RO	X	TX_STAT_PRESENT
12	RO	X	RX_STAT_IDLE
11	RW	0x0	RX_DATA_EN
10	RW	0x0	RX_DATA_READY
9:8	RW	0x3	RX_SLEEP
7	RW	0x0	TX_DATA_EN
6	RW	0x0	TX_DATA_READY
5:4	RW	0x3	TX_SLEEP
3	RW	0x0	CKBUFPD_OVRD
2	RW	0x0	CKBUFPD
1	RW	0x0	IDDQ_OVRD
0	RW	0x0	IDDQ

23.9.10.26 XUSB_PADCTL_IOPHY_MISC_PAD1_CTL_1_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x000aX330 (0bxxxx0000000010100xxx001100110000)

Bit	R/W	Reset	Description
27	RW	0x0	RX_PWR_OVRD
26	RW	0x0	TX_PWR_OVRD
25	RW	0x0	RATE_MODE_OVRD
24	RW	0x0	RATE_MODE
23:22	RW	0x0	RX_DIV
21:20	RW	0x0	TX_DIV
19:18	RW	0x2	RX_RATE
17:16	RW	0x2	TX_RATE
15	RW	0x0	TX_RDET
13	RO	X	TX_STAT_PRESENT
12	RO	X	RX_STAT_IDLE
11	RW	0x0	RX_DATA_EN
10	RW	0x0	RX_DATA_READY
9:8	RW	0x3	RX_SLEEP
7	RW	0x0	TX_DATA_EN
6	RW	0x0	TX_DATA_READY
5:4	RW	0x3	TX_SLEEP

Bit	R/W	Reset	Description
3	RW	0x0	CKBUFPD_OVRD
2	RW	0x0	CKBUFPD
1	RW	0x0	IDDQ_OVRD
0	RW	0x0	IDDQ

23.9.10.27 XUSB_PADCTL_IOPHY_MISC_PAD0_CTL_2_0

Offset: 0x68 | Read/Write: R/W | Reset: 0xX0000000 (0bxx000x00xxxxxxxxx00000000000000)

Bit	R/W	Reset	Description
31:30	RO	X	SPARE_OUT
29:28	RW	0x0	SPARE_IN
27	RW	0x0	TEST_EN
25	RW	0x0	PRBS_CHK_EN
24	RW	0x0	PRBS_ERROR
13	RW	0x0	RX_CDR_RESET
12	RW	0x0	TX_SYNC
11	RW	0x0	FED_LOOP
10:8	RW	0x0	TX_DATA_MODE
7	RW	0x0	FEA_LOOP
6:4	RW	0x0	FEA_MODE
3	RW	0x0	NEA_LOOP
2	RW	0x0	NED_LOOP
1:0	RW	0x0	NED_MODE

23.9.10.28 XUSB_PADCTL_IOPHY_MISC_PAD1_CTL_2_0

Offset: 0x6c | Read/Write: R/W | Reset: 0xX0000000 (0bxx000x00xxxxxxxxx00000000000000)

Bit	R/W	Reset	Description
31:30	RO	X	SPARE_OUT
29:28	RW	0x0	SPARE_IN
27	RW	0x0	TEST_EN
25	RW	0x0	PRBS_CHK_EN
24	RW	0x0	PRBS_ERROR
13	RW	0x0	RX_CDR_RESET
12	RW	0x0	TX_SYNC
11	RW	0x0	FED_LOOP
10:8	RW	0x0	TX_DATA_MODE
7	RW	0x0	FEA_LOOP
6:4	RW	0x0	FEA_MODE
3	RW	0x0	NEA_LOOP

Bit	R/W	Reset	Description
2	RW	0x0	NED_LOOP
1:0	RW	0x0	NED_MODE

23.9.10.29 XUSB_PADCTL_IOPHY_MISC_PAD0_CTL_3_0

Offset: 0x70 | Read/Write: R/W | Reset: 0x00040200 (0b00000000000000010000000010xxxx0000)

Bit	Reset	Description
31:20	0x0	CDR_TEST
19	0x0	RX_IDLE_MODE_OVRD
18	0x1	RX_IDLE_MODE
17	0x0	RX_IDLE_BYP
16	0x0	TX_RDET_BYP
15:14	0x0	RX_IDLE_T
13:12	0x0	TX_RDET_T
11:8	0x2	TX_SEL_LOAD
3:0	0x0	MISC_CNTL

23.9.10.30 XUSB_PADCTL_IOPHY_MISC_PAD1_CTL_3_0

Offset: 0x74 | Read/Write: R/W | Reset: 0x00040200 (0b00000000000000010000000010xxxx0000)

Bit	Reset	Description
31:20	0x0	CDR_TEST
19	0x0	RX_IDLE_MODE_OVRD
18	0x1	RX_IDLE_MODE
17	0x0	RX_IDLE_BYP
16	0x0	TX_RDET_BYP
15:14	0x0	RX_IDLE_T
13:12	0x0	TX_RDET_T
11:8	0x2	TX_SEL_LOAD
3:0	0x0	MISC_CNTL

23.9.10.31 XUSB_PADCTL_IOPHY_MISC_PAD0_CTL_4_0

Offset: 0x78 | Read/Write: R/W | Reset: 0xXX000XX0 (0bx100x0000000xxxxxx0000x000x0xxxx)

Bit	R/W	Reset	Description
31	RO	X	AUX_RX_STAT_IDLE
30	RW	0x1	AUX_RX_IDLE_MODE
29	RW	0x0	AUX_RX_IDLE_EN
28	RW	0x0	AUX_RX_TERM_EN
27	RO	X	AUX_TX_STAT_PRESENT
26	RW	0x0	AUX_TX_RDET_CLK_EN

Bit	R/W	Reset	Description
25	RW	0x0	AUX_TX_RDET_EN
24	RW	0x0	AUX_TX_TERM_EN
23	RW	0x0	AUX_MODE_OVRD
22	RW	0x0	AUX_HOLD_EN
21	RW	0x0	AUX_IDDQ_OVRD
20	RW	0x0	AUX_IDDQ
13	RW	0x0	TX_BYP_OVRD
12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_EN
10	RW	0x0	RX_BYP_DIR
9	RO	X	RX_BYP_IN
8	RW	0x0	RX_BYP_OUT
7	RW	0x0	TX_BYP_EN
6	RW	0x0	TX_BYP_DIR
5	RO	X	TX_BYP_IN
4	RW	0x0	TX_BYP_OUT

23.9.10.32 XUSB_PADCTL_IOPHY_MISC_PAD1_CTL_4_0

Offset: 0x7c | Read/Write: R/W | Reset: 0xXX000XX0 (0bx100x0000000xxxxxx00000x00x0xxxx)

Bit	R/W	Reset	Description
31	RO	X	AUX_RX_STAT_IDLE
30	RW	0x1	AUX_RX_IDLE_MODE
29	RW	0x0	AUX_RX_IDLE_EN
28	RW	0x0	AUX_RX_TERM_EN
27	RO	X	AUX_TX_STAT_PRESENT
26	RW	0x0	AUX_TX_RDET_CLK_EN
25	RW	0x0	AUX_TX_RDET_EN
24	RW	0x0	AUX_TX_TERM_EN
23	RW	0x0	AUX_MODE_OVRD
22	RW	0x0	AUX_HOLD_EN
21	RW	0x0	AUX_IDDQ_OVRD
20	RW	0x0	AUX_IDDQ
13	RW	0x0	TX_BYP_OVRD
12	RW	0x0	RX_BYP_MODE
11	RW	0x0	RX_BYP_EN
10	RW	0x0	RX_BYP_DIR
9	RW	0x0	RX_BYP_OUT
8	RO	X	RX_BYP_IN

Bit	R/W	Reset	Description
7	RW	0x0	TX_BYP_EN
6	RW	0x0	TX_BYP_DIR
5	RO	X	TX_BYP_IN
4	RW	0x0	TX_BYP_OUT

23.9.10.33 XUSB_PADCTL_IOPHY_MISC_PAD0_CTL_5_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x0000X0XX (0bxxxxxxxxxxxxxxxxxxxxxxxx00xx00xx0)

Bit	R/W	Reset	Description
15:12	RO	X	RX_QEYE_OUT
8	RW	0x0	RX_QEYE_EN
7	RW	0x0	EOM_EN
5	RO	X	EOM_TRAIN_DONE
4	RW	0x0	EOM_TRAIN_EN
3	RW	0x0	DFE_RESET
1	RO	X	DFE_TRAIN_DONE
0	RW	0x0	DFE_TRAIN_EN

23.9.10.34 XUSB_PADCTL_IOPHY_MISC_PAD1_CTL_5_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x0000X0XX (0bxxxxxxxxxxxxxxxxxxxxxxxx00xx00xx0)

Bit	R/W	Reset	Description
15:12	RO	X	RX_QEYE_OUT
8	RW	0x0	RX_QEYE_EN
7	RW	0x0	EOM_EN
5	RO	X	EOM_TRAIN_DONE
4	RW	0x0	EOM_TRAIN_EN
3	RW	0x0	DFE_RESET
1	RO	X	DFE_TRAIN_DONE
0	RW	0x0	DFE_TRAIN_EN

23.9.10.35 XUSB_PADCTL_IOPHY_MISC_PAD0_CTL_6_0

Offset: 0x88 | Read/Write: R/W | Reset: 0xXX000000 (0bxxxxxxxx00000000000000000000000)

Bit	R/W	Reset	Description
31:24	RO	X	MISC_OUT
23:16	RW	0x0	MISC_OUT_SEL
15:0	RW	0x0	MISC_TEST

23.9.10.36 XUSB_PADCTL_IOPHY_MISC_PAD1_CTL_6_0

Offset: 0x8c | Read/Write: R/W | Reset: 0xXX000000 (0bxxxxxxxx00000000000000000000000)

Bit	R/W	Reset	Description
31:24	RO	X	MISC_OUT
23:16	RW	0x0	MISC_OUT_SEL
15:0	RW	0x0	MISC_TEST

23.9.10.37 XUSB_PADCTL_USB2_OTG_PAD0_CTL_0_0

USB2 OTG, HSIC, ICUSB Control Settings

OTGPAD0, CTL0 static settings

One set per pad (except the BIAS pad which is just one set)

Offset: 0x90 | Read/Write: R/W | Reset: 0x003808a0 (0bxxxxxxxx00111000000100010100000)

Bit	Reset	Description
23	0x0	LSBIAS_SEL
22	0x0	DISCON_DETECT_METHOD
21	0x1	PD_ZI
20	0x1	PD2
19	0x1	PD
18	0x0	TERM_EN
17:16	0x0	LS_FSLEW
15:14	0x0	LS_RSLEW
13:12	0x0	FS_SLEW
11:6	0x22	HS_SLEW
5:0	0x20	HS_CURR_LEVEL

23.9.10.38 XUSB_PADCTL_USB2_OTG_PAD1_CTL_0_0

OTGPAD1, CTL0 static settings

Offset: 0x94 | Read/Write: R/W | Reset: 0x003808a0 (0bxxxxxxxx00111000000100010100000)

Bit	Reset	Description
23	0x0	LSBIAS_SEL
22	0x0	DISCON_DETECT_METHOD
21	0x1	PD_ZI
20	0x1	PD2
19	0x1	PD
18	0x0	TERM_EN
17:16	0x0	LS_FSLEW
15:14	0x0	LS_RSLEW

Bit	Reset	Description
13:12	0x0	FS_SLEW
11:6	0x22	HS_SLEW
5:0	0x20	HS_CURR_LEVEL

23.9.10.39 XUSB_PADCTL_USB2_OTG_PAD0_CTL_1_0

OTGPAD0, CTL1 static settings

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxx0000000000100)

Bit	Reset	Description
12:11	0x0	RPU_RANGE_ADJ
10:9	0x0	HS_IREF_CAP
8:7	0x0	SPARE
6:3	0x0	TERM_RANGE_ADJ
2	0x1	PD_DR
1	0x0	PD_DISC_FORCE_POWERUP
0	0x0	PD_CHRP_FORCE_POWERUP

23.9.10.40 XUSB_PADCTL_USB2_OTG_PAD1_CTL_1_0

OTGPAD1, CTL1 static settings

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxx0000000000100)

Bit	Reset	Description
12:11	0x0	RPU_RANGE_ADJ
10:9	0x0	HS_IREF_CAP
8:7	0x0	SPARE
6:3	0x0	TERM_RANGE_ADJ
2	0x1	PD_DR
1	0x0	PD_DISC_FORCE_POWERUP
0	0x0	PD_CHRP_FORCE_POWERUP

23.9.10.41 XUSB_PADCTL_USB2_BIAS_PAD_CTL_0_0

Offset: 0xa0 | Read/Write: R/W | Reset: 0x00003000 (0bxxxxxxxxxxxxxxxx00011000000000000)

Bit	Reset	Description
16:14	0x0	ADJRPU
13	0x1	PD_TRK
12	0x1	PD
11:9	0x0	TERM_OFFSET
8:7	0x0	VBUS_LEVEL
6:5	0x0	HS_CHIRP_LEVEL

Bit	Reset	Description
4:2	0x0	HS_DISCON_LEVEL
1:0	0x0	HS_SQUELCH_LEVEL

23.9.10.42 XUSB_PADCTL_USB2_BIAS_PAD_CTL_1_0

BIASPAD, CTL1 static settings

Offset: 0xa4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TCTRL
15:0	X	RCTRL

23.9.10.43 XUSB_PADCTL_HSIC_PAD0_CTL_0_0

HSIC PAD0, CTL0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00008888 (0bxxxxxxxxxxxxx0000100010001000 10001000) | Default: 0x00000000

Bit	Reset	SW Default	Description
19:16	0x0	_NONE_	HSIC_OPT
15:12	0x8	0x0	TX_SLEWN
11:8	0x8	0x0	TX_SLEWP
7:4	0x8	_NONE_	TX_RTUNEN
3:0	0x8	_NONE_	TX_RTUNEP

23.9.10.44 XUSB_PADCTL_HSIC_PAD1_CTL_0_0

HSIC PAD1, CTL0

Offset: 0xac | Read/Write: R/W | Reset: 0x00008888 (0bxxxxxxxxxxxxx0000100010001000 10001000) | Default: 0x00000000

Bit	Reset	SW Default	Description
19:16	0x0	_NONE_	HSIC_OPT
15:12	0x8	0x0	TX_SLEWN
11:8	0x8	0x0	TX_SLEWP
7:4	0x8	_NONE_	TX_RTUNEN
3:0	0x8	_NONE_	TX_RTUNEP

23.9.10.45 XUSB_PADCTL_HSIC_PAD0_CTL_1_0

HSIC PAD0, CTL1

Offset: 0xb0 | Read/Write: R/W | Reset: 0x000004b8 (0bxxxxxxxxxxxxxxxxxxxxxx10010111000)

Bit	Reset	Description
10	0x1	RPU_STROBE
9	0x0	RPU_DATA

Bit	Reset	Description
8	0x0	RPD_STROBE
7	0x1	RPD_DATA
6	0x0	LPBK
5	0x1	PD_ZI
4	0x1	PD_RX
3	0x1	PD_TRX
2	0x0	PD_TX
1	0x0	IDDQ
0	0x0	AUTO_TERM_EN

23.9.10.46 XUSB_PADCTL_HSIC_PAD1_CTL_1_0

HSIC PAD1, CTL1

Offset: 0xb4 | Read/Write: R/W | Reset: 0x000004b8 (0bxxxxxxxxxxxxxxxxxxxxxx10010111000)

Bit	Reset	Description
10	0x1	RPU_STROBE
9	0x0	RPU_DATA
8	0x0	RPD_STROBE
7	0x1	RPD_DATA
6	0x0	LPBK
5	0x1	PD_ZI
4	0x1	PD_RX
3	0x1	PD_TRX
2	0x0	PD_TX
1	0x0	IDDQ
0	0x0	AUTO_TERM_EN

23.9.10.47 XUSB_PADCTL_HSIC_PAD0_CTL_2_0

HSIC PAD0, CTL2

Offset: 0xb8 | Read/Write: R/W | Reset: 0xXXXX0022 (0bxxxxxxxxxxxxxxxxxxxxxxxx00100010)

Bit	R/W	Reset	Description
31:16	RO	X	CALIOUT
7:4	RW	0x2	RX_STROBE_TRIM
3:0	RW	0x2	RX_DATA_TRIM

23.9.10.48 XUSB_PADCTL_HSIC_PAD1_CTL_2_0

HSIC PAD1, CTL2

Offset: 0xbc | Read/Write: R/W | Reset: 0xXXXX0022 (0bxxxxxxxxxxxxxxxxxxxxxxxx00100010)

Bit	R/W	Reset	Description
31:16	RO	X	CALIOUT
7:4	RW	0x2	RX_STROBE_TRIM
3:0	RW	0x2	RX_DATA_TRIM

23.9.10.49 XUSB_PADCTL_ULPI_LINK_TRIM_CONTROL_0

ULPI LINK and NULL Mode Trimmer Controls

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx0000000000xxxxx00x00000000)

Bit	Reset	Description
25	0x0	CTL_SEL_DEL1
24	0x0	CTL_SEL_DEL0
23:16	0x0	CTL_TRIM_VAL
10	0x0	DAT_SEL_DEL1
9	0x0	DAT_SEL_DEL0
7:0	0x0	DAT_TRIM_VAL

23.9.10.50 XUSB_PADCTL_ULPI_NULL_CLK_TRIM_CONTROL_0

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx00000xxx00000)

Bit	Reset	Description
12:8	0x0	NULL_LBKCLK_TRIM_VAL
4:0	0x0	NULL_CLKOUT_TRIM_VAL

23.9.10.51 XUSB_PADCTL_HSIC_STRB_TRIM_CONTROL_0

HSIC STROBE Trimmer Control

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx000001)

Bit	Reset	Description
5:0	0x1	STRB_TRIM_VAL

23.9.10.52 XUSB_PADCTL_WAKE_CTRL_0

Wake Logic AUX RDET CLK FORCE ON

Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	PORT1_FORCE_TX_RDET_CLK_ENABLE
0	0x0	PORT0_FORCE_TX_RDET_CLK_ENABLE

23.9.10.53 XUSB_PADCTL_PM_SPARE_0

PAD MACRO Spare Bits

Offset: 0xd0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11	0x0	HSIC_PM_SPARE_BIT3
10	0x0	HSIC_PM_SPARE_BIT2
9	0x0	HSIC_PM_SPARE_BIT1
8	0x0	HSIC_PM_SPARE_BIT0
7	0x0	ULPI_PM_SPARE_BIT3
6	0x0	ULPI_PM_SPARE_BIT2
5	0x0	ULPI_PM_SPARE_BIT1
4	0x0	ULPI_PM_SPARE_BIT0
3	0x0	OTG_PM_SPARE_BIT3
2	0x0	OTG_PM_SPARE_BIT2
1	0x0	OTG_PM_SPARE_BIT1
0	0x0	OTG_PM_SPARE_BIT0

23.9.11 XUSB PCI Config Registers

23.9.11.1 T_XUSB_CFG_0

Offset: 0x00 | Read/Write: R

Bits	Reset	R/W	Description
31:16	E16h	R	T_XUSB_CFG_0_DEVICE_ID_UNIT: E16h: DEVICE_ID_UNIT_XUSB (default)
15:0	10DEh	R	T_XUSB_CFG_0_VENDOR_ID: 10DEh: VENDOR_ID_NVIDIA (default)

23.9.11.2 T_XUSB_CFG_1

FPCI Configuration Register

Offset: 0x04 | Read/Write: R/W

Bits	Reset	R/W	Description
26:25	0	R	T_XUSB_CFG_1_DEVSEL_TIMING: 0h: DEVSEL_TIMING_FAST (default) 1h: DEVSEL_TIMING_MEDIUM 2h: DEVSEL_TIMING_SLOW
31	0	R	T_XUSB_CFG_1_DETECTED_PERR: 0h: DETECTED_PERR_NOT_ACTIVE (default) 1h: DETECTED_PERR_ACTIVE 1h: DETECTED_PERR_CLEAR
30	0	R	T_XUSB_CFG_1_SIGNALED_SERR: 0h: SIGNALED_SERR_NOT_ACTIVE (default) 1h: SIGNALED_SERR_ACTIVE 1h: SIGNALED_SERR_CLEAR

Bits	Reset	R/W	Description
18:11	0	R	Reserved
29	0	R	T_XUSB_CFG_1_RECEIVED_MASTER: 0h: RECEIVED_MASTER_NO_ABORT (default) 1h: RECEIVED_MASTER_ABORT 1h: RECEIVED_MASTER_CLEAR
28	0	R	T_XUSB_CFG_1_RECEIVED_TARGET: 0h: RECEIVED_TARGET_NO_ABORT (default) 1h: RECEIVED_TARGET_ABORT 1h: RECEIVED_TARGET_CLEAR
27	0	R	T_XUSB_CFG_1_SIGNALED_TARGET: 0h: SIGNALED_TARGET_NO_ABORT (default) 1h: SIGNALED_TARGET_ABORT 1h: SIGNALED_TARGET_CLEAR
24	0	R	T_XUSB_CFG_1_MASTER_DATA_PERR: 0h: MASTER_DATA_PERR_NOT_ACTIVE (default) 1h: MASTER_DATA_PERR_ACTIVE 1h: MASTER_DATA_PERR_CLEAR
23	1h	R	T_XUSB_CFG_1_FAST_BACK2BACK: 0h: FAST_BACK2BACK_INCAPABLE 1h: FAST_BACK2BACK_CAPABLE (default)
22	0	R	Reserved
21	1h	R	T_XUSB_CFG_1_66MHZ: 0h: 66MHZ_INCAPABLE 1h: 66MHZ_CAPABLE (default)
20	1h	R	T_XUSB_CFG_1_CAPLIST: 0h: CAPLIST_NOT_PRESENT 1h: CAPLIST_PRESENT (default)
19	None	R	T_XUSB_CFG_1_INTR_STATUS: 0h: INTR_STATUS_0 1h: INTR_STATUS_1
10	0	R/W	T_XUSB_CFG_1_INTR_DISABLE: 0h: INTR_DISABLE_ON (default) 1h: INTR_DISABLE_OFF
9	0	R	T_XUSB_CFG_1_BACK2BACK: 0h: BACK2BACK_DISABLED (default) 1h: BACK2BACK_ENABLED
8	0	R	T_XUSB_CFG_1_SERR: 0h: SERR_DISABLED (default) 1h: SERR_ENABLED
7	0	R	T_XUSB_CFG_1_STEP: 0h: STEP_DISABLED (default) 1h: STEP_ENABLED
6	0	R	T_XUSB_CFG_1_PERR: 0h: PERR_DISABLED (default) 1h: PERR_ENABLED
5	0	R	T_XUSB_CFG_1_PALETTE_SNOOP: 0h: PALETTE_SNOOP_DISABLED (default) 1h: PALETTE_SNOOP_ENABLED
4	0	R	T_XUSB_CFG_1_WRITE_AND_INVAL: 0h: WRITE_AND_INVAL_DISABLED (default) 1h: WRITE_AND_INVAL_ENABLED
3	0	R	T_XUSB_CFG_1_SPECIAL_CYCLE: 0h: SPECIAL_CYCLE_DISABLED (default) 1h: SPECIAL_CYCLE_ENABLED

Bits	Reset	R/W	Description
2	0	R/W	T_XUSB_CFG_1_BUS_MASTER: 0h: BUS_MASTER_DISABLED (default) 1h: BUS_MASTER_ENABLED
1	0	R/W	T_XUSB_CFG_1_MEMORY_SPACE: 0h: MEMORY_SPACE_DISABLED (default) 1h: MEMORY_SPACE_ENABLED
0	0	R/W	T_XUSB_CFG_1_IO_SPACE: 0h: IO_SPACE_DISABLED (default) 1h: IO_SPACE_ENABLED

23.9.11.3 T_XUSB_CFG_2

PCI Revision ID and Class Code Register

Offset: 0x08 | Read/Write: R

Bits	Reset	R/W	Description
31:24	Ch	R	T_XUSB_CFG_2_BASE_CLASS: The CLASS_CODE bits identify the generic function of the device and (in some cases) a specific register-level programming interface. The register is broken into three byte-size fields. The upper byte (at offset 0BH) is a base class code which broadly classifies the type of function the device performs. The middle-byte (at offset 0BH) is a sub-class code which identifies more specifically the function of the device. The lower byte (at offset 09H) identifies a specific register-level programming interface (if any) so that device independent software can interact with the device. The Class Code and Revision ID are defined by parameters per block. Ch: BASE_CLASS_SBC (default)
23:16	3h	R	T_XUSB_CFG_2_SUB_CLASS: 3h: SUB_CLASS_XUSB (default)
15:8	30h	R	T_XUSB_CFG_2_PROG_IF: 30h: PROG_IF_XHCI (default)
7:0	A1h	R	T_XUSB_CFG_2_REVISION_ID: The REVISION_ID bits specify a device specific revision identifier. The value is chosen by the vendor. Zero is an acceptable value. This field should be viewed as a vendor defined extension to the DEVICE_ID. A1h: REVISION_ID_VAL (default) A1h: REVISION_ID_A01

23.9.11.4 T_XUSB_CFG_3

PCI Configuration Register

Offset: 0x0C | Read/Write: R

Bits	Reset	R/W	Description
31:24	0	R	Reserved
23	0	R	T_XUSB_CFG_3_HEADER_TYPE_FUNC: 0h: HEADER_TYPE_FUNC_SINGLE (default) 1h: HEADER_TYPE_FUNC_MULTI
22:16	0	R	T_XUSB_CFG_3_HEADER_TYPE_DEVICE: The HEADER_TYPE bits identify the layout of the bytes 10h through 3Fh in configuration space and also whether or not the device contains multiple functions. Bit 7 in this register is used to identify a multi-function device. If the bit is 0, then the device is single function. If the bit is 1, then the device has multiple functions. Bits 6 through 0 specify the layout of bytes 10h through 3Fh. The LATENCY_TIMER and HEADER_TYPE are defined by parameters per block. 0h: HEADER_TYPE_DEVICE_NON_BRIDGE (default) 1h: HEADER_TYPE_DEVICE_P2P_BRIDGE
15:11	0	R	T_XUSB_CFG_3_LATENCY_TIMER:

Bits	Reset	R/W	Description
			The LATENCY_TIMER bits contain, in units of PCI bus clocks, the value of the Latency Timer for this PCI bus master. This register must be implemented as writable by any master that can burst more than two data phases. This register may be implemented as read-only for devices that burst two or fewer data phases, but the hardwired value must be limited to 16 or less. A typical implementation would be to build the five high-order bits (leaving the bottom three as read-only), resulting in a timer granularity of eight clocks. At reset, the register should be set to 0 (if programmable). LATENCY_TIMER bits are writable. 0h: LATENCY_TIMER_0_CLOCKS (default) 1h: LATENCY_TIMER_8_CLOCKS 1Eh: LATENCY_TIMER_240_CLOCKS 1Fh: LATENCY_TIMER_248_CLOCKS
10:8	0	R	Reserved
7:0	0	R	T_XUSB_CFG_3_CACHE_LINE_SIZE: 0h: CACHE_LINE_SIZE_0 (default) 20h: CACHE_LINE_SIZE_32 40h: CACHE_LINE_SIZE_64

23.9.11.5 T_XUSB_CFG_4

PCI Configuration Register

Offset: 0x10 | Read/Write: R/W

Bits	Reset	R/W	Description
31:15	0	R/W	T_XUSB_CFG_4_BASE_ADDRESS: The BASE_ADDRESS bits contain the base address of the device. The number of upper bits that a device actually implements depends on how much of the address space the device will respond to. A device that wants a 1 MB memory address space (using a 32-bit base address register) would build the top 12 bits of the address register, hardwiring the other bits to 0. Power-up software can determine how much address space the device required by writing a value of all 1's to the register and then reading the value back. The device will return 0's in all don't-care address bits, effectively specifying the address space required. 0h: BASE_ADDRESS_DEFAULT (default)
14:4	0	R	T_XUSB_CFG_4_BAR_SIZE_32KB: 0h: BAR_SIZE_32KB_RSVD (default)
3	1h	R	T_XUSB_CFG_4_PREFETCHABLE: 0h: PREFETCHABLE_NOT 1h: PREFETCHABLE_MERGABLE (default)
2:1	2h	R	T_XUSB_CFG_4_ADDRESS_TYPE: The ADDRESS_TYPE bits contain the type of the Base Address. It can be 32 bits, 20 bits, or 64 bits wide. 0h: ADDRESS_TYPE_32_BIT 2h: ADDRESS_TYPE_64_BIT (default)
0	0	R	T_XUSB_CFG_4_SPACE_TYPE: The SPACE_TYPE bit indicates whether the register maps into Memory or I/O space. 0h: SPACE_TYPE_MEMORY (default) 1h: SPACE_TYPE_IO

23.9.11.6 T_XUSB_CFG_5

Offset: 0x14 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_CFG_5_BASE_ADDRESSHI: 0h: BASE_ADDRESSHI_DEFAULT (default)

23.9.11.7 T_XUSB_CFG_6

Offset: 0x18-0x28 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_CFG_6_RSVD: 0h: RSVD_00 (default)

23.9.11.8 T_XUSB_CFG_11

PCI Configuration Register

The SUBSYSTEM_VENDOR_ID bits and SUBSYSTEM_ID bits are used to uniquely identify the add-in board or subsystem where the device resides. When the device is on the motherboard, there is no serial ROM and the registers both initialize to NONE. The motherboard BIOS must set the values of the Subsystem ID and Subsystem Vendor ID by writing the proper values to the SUBSYSTEM_VENDOR_ID and SUBSYSTEM_ID bits in the PCI_T_16 register (NOT PCI_T_11).

Offset: 0x2c | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_CFG_11_SUBSYSTEM_ID: 0h: SUBSYSTEM_ID_NONE (default)
15:0	0	R	T_XUSB_CFG_11_SUBSYSTEM_VENDOR_ID: 0h: SUBSYSTEM_VENDOR_ID_NONE (default)

23.9.11.9 T_XUSB_CFG_12

PCI Configuration Register

Offset: 0x30 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_CFG_12_RESERVED: 0h: RESERVED_0 (default)

23.9.11.10 T_XUSB_CFG_13

PCI Capability Pointer Register

Offset: 0x34 | Read/Write: R

Bits	Reset	R/W	Description
31:8	0	R	Reserved
7:0	44h	R	T_XUSB_CFG_13_CAP_PTR: The CAP_PTR bits indicate the offset into configuration space where the capabilities list begins. This always points to 0x44 where at least the PCI-PM registers are expected to reside. 44h: CAP_PTR_PMCAP C0h: CAP_PTR_MSI 70h: CAP_PTR_MSIX DCh: CAP_PTR_MMAP 44h: CAP_PTR_DEFAULT (default)

23.9.11.11 T_XUSB_CFG_14

PCI Configuration Register

Offset: 0x38 | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_CFG_14_RESERVED: 0h: RESERVED_0 (default)

23.9.11.12 T_XUSB_CFG_15

PCI Configuration Register

Offset: 0x3c | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	T_XUSB_CFG_15_MAX_LAT: The MAX_LAT bits contain the maximum time the device requires to gain access to the CPI bus. This read-only register is used to specify the device's desired settings for Latency Timer values. The value specifies a period of time in units of 1/4 microseconds. Values of 0 indicate that the device has no major requirements for the settings of Latency Timers. MAX_LAT is nonzero. The INTR_PIN, MIN_GNT, and MAX_LAT are configurable per block. For a 64-byte buffer with two 32-byte sections, the maximum tolerable latency for the XHCI once a large XUSB ISO transaction has begun is about 23 µs. The latency timer is hardwired to 20 µs. This value only applies in External PCI operation. 0h: MAX_LAT_NO_REQUIREMENTS (default) 14h: MAX_LAT_5US 50h: MAX_LAT_20US
23:16	0	R	T_XUSB_CFG_15_MIN_GNT: The MIN_GNT bits contain the length of the burst period a device needs assuming a clock rate of 33 MHz. This read-only register is used to specify the device's desired settings for Latency Timer values. The value specifies a period of time in units of 1/4 microsecond. Values of 0 indicate that the device has no major requirements for the settings of Latency Timers. MIN_GNT is nonzero. 0h: MIN_GNT_NO_REQUIREMENTS (default) 1h: MIN_GNT_240NS
15:8	1h	R	T_XUSB_CFG_15_INTR_PIN: The INTR_PIN bits contain the interrupt pin the device (or device function) uses. A value of 1 corresponds to INTA#. A value of 2 corresponds to INTB#. A value of 3 corresponds to INTC#. A value of 4 corresponds to INTD#. Devices (or device functions) that do not use an interrupt pin must put a 0 in this register. 0h: INTR_PIN_NONE 1h: INTR_PIN_INTA (default) 2h: INTR_PIN_INTB 3h: INTR_PIN_INTC 4h: INTR_PIN_INTD
7:0	0	R/W	T_XUSB_CFG_15_INTR_LINE: The INTR_LINE bits contain the interrupt routing information. The register is read/write and must be implemented by any device (or device function) that uses an interrupt pin. POST software will write the routing information into this register as it initializes and configures the system. The value in this register tells which input of the system interrupt controller(s) the device's interrupt pin is connected to. Device drivers and operating systems can use this information to determine priority and vector information. INTR_LINE is initialized to 0xff (no connection) at reset. Some PCI BIOS' cannot handle aliased INTR_LINEs. Some PCI BIOS' cannot handle INTR_LINE initialized to 0xff. 0h: INTR_LINE_IRQ0 (default) 1h: INTR_LINE_IRQ1 Fh: INTR_LINE_IRQ15 FFh: INTR_LINE_UNKNOWN

23.9.11.13 T_XUSB_CFG_16

Backdoor register write for updating subsystem ID/vendor ID.

Offset: 0x40 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R/W	T_XUSB_CFG_16_SUBSYSTEM_ID: 0h: SUBSYSTEM_ID_NONE (default)
15:0	0	R/W	T_XUSB_CFG_16_SUBSYSTEM_VENDOR_ID: 0h: SUBSYSTEM_VENDOR_ID_NONE (default)

23.9.11.14 T_XUSB_CFG_17

Offset: 0x44 | Read/Write: R

Bits	Reset	R/W	Description
31	1h	R	T_XUSB_CFG_17_D3CPME_SUPPORT: 1h: D3CPME_SUPPORT_YES 0h: D3CPME_SUPPORT_NO
30	1h	R	T_XUSB_CFG_17_D3HPME_SUPPORT: 1h: D3HPME_SUPPORT_YES 0h: D3HPME_SUPPORT_NO
29	0	R	T_XUSB_CFG_17_D2PME_SUPPORT: 1h: D2PME_SUPPORT_YES 0h: D2PME_SUPPORT_NO
28	0	R	T_XUSB_CFG_17_D1PME_SUPPORT: 1h: D1PME_SUPPORT_YES 0h: D1PME_SUPPORT_NO
27	1h	R	T_XUSB_CFG_17_D0PME_SUPPORT: 1h: D0PME_SUPPORT_YES 0h: D0PME_SUPPORT_NO
26	0	R	T_XUSB_CFG_17_D2_SUPPORT: 0h: D2_SUPPORT_NO 1h: D2_SUPPORT_YES
25	0	R	T_XUSB_CFG_17_D1_SUPPORT: 0h: D1_SUPPORT_NO 1h: D1_SUPPORT_YES
24:22	0	R	T_XUSB_CFG_17_AUXCUR: 0h: AUXCUR_SELF 1h: AUXCUR_55MA 2h: AUXCUR_100MA 3h: AUXCUR_160MA 4h: AUXCUR_220MA 5h: AUXCUR_270MA 6h: AUXCUR_320MA 7h: AUXCUR_375MA
21	0	R	T_XUSB_CFG_17_DSI: 0h: DSI_NONE 1h: DSI_NEEDED
20	0	R	T_XUSB_CFG_17_RSVD: 0h: RSVD_0
19	0	R	T_XUSB_CFG_17_PMECLK: 0h: PMECLK_NOT_REQUIRED 1h: PMECLK_REQUIRED
18:16	3h	R	T_XUSB_CFG_17_VER: 3h: VER_1P2

Bits	Reset	R/W	Description
15:8	C0h	R	T_XUSB_CFG_17_NEXT_PTR: C0h: NEXT_PTR_MSI 70h: NEXT_PTR_MSIX DCh: NEXT_PTR_MMAP 0h: NEXT_PTR_NULL C0h: NEXT_PTR_DEFAULT (default)
7:0	1h	R	T_XUSB_CFG_17_CAP: 1h: CAP_PCIPM

23.9.11.15 T_XUSB_CFG_18_PMCSR

Offset: 0x48 | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	Reserved
23:16	0	R	T_XUSB_CFG_18_PMCSR_BSE_RSVD: 0h: BSE_RSVD_00
15	0	RW1C	T_XUSB_CFG_18_PMCSR_PMESTATUS: 0h: PMESTATUS_NOT_PENDING (default) 1h: PMESTATUS_PENDING 1h: PMESTATUS_CLEAR
14:13	0	R	T_XUSB_CFG_18_PMCSR_DSCALE: 0h: DSCALE_INIT
12:9	0	R	T_XUSB_CFG_18_PMCSR_DSEL: 0h: DSEL_INIT
8	0	R/W	T_XUSB_CFG_18_PMCSR_PME: 1h: PME_ENABLE 0h: PME_DISABLE (default)
7:4	0	R	T_XUSB_CFG_18_PMCSR_RSVD1: 0h: RSVD1_00
3	1h	R	T_XUSB_CFG_18_PMCSR_NSR: 1h: NSR_NORESET 0h: NSR_RESET
2	0	R	T_XUSB_CFG_18_PMCSR_RSVD0: 0h: RSVD0_0
1:0	0	R/W	T_XUSB_CFG_18_PMCSR_PWRSTATE: 0h: PWRSTATE_D0 (default) 1h: PWRSTATE_D1 2h: PWRSTATE_D2 3h: PWRSTATE_D3H

23.9.11.16 T_XUSB_CFG_24

XUSB XHCI Configuration Control

Offset: 0x60 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15:14	0	R	T_XUSB_CFG_24_RSVDP: 0h: RSVDP_VALUE
13:8	20h	R/W	T_XUSB_CFG_24_FLADJ: Frame Length Adjustment Register. This register is in the auxiliary power well. This feature is used to adjust any offset from the clock source that generates the clock that drives the SOF counter. When a new value is written into these six bits,

Bits	Reset	R/W	Description
			the length of the frame is adjusted. Its initial programmed value is system dependent based on the accuracy of hardware USB clock and is initialized by system BIOS. This register should only be modified when the HChalted bit in the USBSTS register is a one. Changing value of this register while the host controller is operating yields undefined results. It should not be reprogrammed by USB system software unless the default or BIOS programmed values are incorrect, or the system is restoring the register while returning from a suspended state. 20h: FLADJ_VALUE (default)
7:0	30h	R	T_XUSB_CFG_24_SBRN: Serial Bus Release Number Register. This register contains the release of the Universal Serial Bus Specification with which this Universal Serial Bus Host Controller module is compliant. 30h: SBRN_VALUE

23.9.11.17 T_XUSB_CFG_EMU_RSVD

MSIX is not part of Tegra 4 devices.

The PCIE and HT capabilities are not required. PCIE capability will be trapped by the emulation only IP - the emulation only IP put the PCIE capability at 80h ~ BBh, and uses BCh ~ BFh, so these regions are reserved for the emulation only IP. (The emulation only IP will mask the pointers to insert the PCIE capability to the config space of the unit.)

Offset: 0x80-0xBB | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	Unknown	R/W	T_XUSB_CFG_EMU_RSVD_FIELD:

23.9.11.18 T_XUSB_MSI_CTRL

MSI Message Control and Capability Register B0h

MSI_CTRL is the main control register for MSI support.

Offset: 0xc0 | Read/Write: R/W

Bits	Reset	R/W	Description
31:25	0	R	Reserved
24	0	R	T_XUSB_MSI_CTRL_VECTOR_MASK_CAP: The VECTOR_MASK_CAP field indicates whether or not the controller supports MSI-per-vector masking. 0h: VECTOR_MASK_CAP_DIS 1h: VECTOR_MASK_CAP_EN 0h: VECTOR_MASK_CAP_DEFAULT (default)
23	1h	R	T_XUSB_MSI_CTRL_64_ADDR_CAP: The 64_ADDR_CAP field indicates whether or not the controller is capable of generating a 64-bit message address. A value of 1 means the controller is capable of generating a 64-bit message address. 0h: 64_ADDR_CAP_DIS 1h: 64_ADDR_CAP_EN 1h: 64_ADDR_CAP_DEFAULT (default)

Bits	Reset	R/W	Description
22:20	0	R/W	T_XUSB_MSI_CTRL_MULT_MSG_ENABLE: System software writes to this field to indicate the number of allocated vectors (less than or equal to the number of vectors requested). The number of vectors is aligned as a power of two. When MSI is enabled, the controller will be allocated at least one vector. 0h: MULT_MSG_ENABLE_1 1h: MULT_MSG_ENABLE_2 2h: MULT_MSG_ENABLE_4 3h: MULT_MSG_ENABLE_8 4h: MULT_MSG_ENABLE_16 5h: MULT_MSG_ENABLE_32 0h: MULT_MSG_ENABLE_DEFAULT (default)
19:17	0	R	T_XUSB_MSI_CTRL_MULT_MSG_CAP: System software reads this field to determine the number of requested vectors. The number of requested vectors must be aligned to a power of two. Values of 6 and 7 in this field are reserved. 0h: MULT_MSG_CAP_1 1h: MULT_MSG_CAP_2 2h: MULT_MSG_CAP_4 3h: MULT_MSG_CAP_8 4h: MULT_MSG_CAP_16 5h: MULT_MSG_CAP_32 0h: MULT_MSG_CAP_DEFAULT (default)
16	0	R/W	T_XUSB_MSI_CTRL_MSI_ENABLE: The MSI_ENABLE field enables the MSI capability. If MSI_ENABLE is written to a 1, the controller is permitted to use MSI to request service and is prohibited from using the legacy interrupt. System configuration software sets this bit to enable MSI. A device driver is prohibited from writing this bit to mask the controller's service request. If this bit is written to a 0, the controller is prohibited from using MSI to request service. 0h: MSI_ENABLE_OFF 1h: MSI_ENABLE_ON 0h: MSI_ENABLE_DEFAULT (default)
15:8	E0h	R	T_XUSB_MSI_CTRL_NEXT_PTR: The NEXT_PTR field identifies the next item in the capabilities list. It is a read-only field. 70h: NEXT_PTR_MSIX DCh: NEXT_PTR_MMAP 44h: NEXT_PTR_PMCAP 0h: NEXT_PTR_NULL E0h: NEXT_PTR_MAILBOX E0h: NEXT_PTR_DEFAULT (default)
7:0	5h	R	T_XUSB_MSI_CTRL_CAP_ID: The CAP_ID field identifies this capability block as the MSI capability block. This is read-only as 0x5. 5h: CAP_ID_MSI (default)

23.9.11.19 T_XUSB_MSI_ADDR1

MSI Message Address Register 84h

MSI_ADDR1 specifies the lower 32 bits to which an MSI memory write transaction should occur.

Offset: 0xc4 | Read/Write: R/W

Bits	Reset	R/W	Description
31:2	0	R/W	T_XUSB_MSI_ADDR1_MSG_ADDR: System-specified message address. When MSI is enabled, this field specifies the Dword-aligned address for the MSI memory write transaction. 0h: MSG_ADDR_DEFAULT (default)
1:0	0	R	Reserved

23.9.11.20 T_XUSB_MSI_ADDR2

MSI Message Upper Address Register 88h

MSI_ADDR2 specifies the upper 32 bits to which an MSI memory write transaction should occur.

Offset: 0xc8 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_MSI_ADDR2_MSG_ADDR: System-specified message address. When MSI is enabled, this field specifies the upper 32 bits of the address for the MSI memory write transaction. The contents of this register only apply when T_XUSB_CFG_MSI_CTRL_64_ADDR_CAP bit is set. 0h: MSG_ADDR_DEFAULT (default)

23.9.11.21 T_XUSB_MSI_DATA

MSI Message Data Register 8Ch

The MSI_DATA register contains the system-specified message.

Offset: 0xcc | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15:0	0	R/W	T_XUSB_MSI_DATA_MSG_DATA: System-specified message When MSI is enabled, the message data is driven onto the lower 16 bits of the MSI memory write. The MULT_MSG_ENABLE field in configuration register 80h specifies the number of low-order message data bits that the XHCI is permitted to modify to generate its system software allocated vectors. 0h: MSG_DATA_DEFAULT (default)

23.9.12 XUSB XHCI Registers

23.9.12.1 T_XUSB_XHCI_CAP_REG0

Offset: 0x00 | Read/Write: R

Bits	Reset	R/W	Description
31:16	Unknown	R	T_XUSB_XHCI_CAP_REG0_HCIVERSION: 100h: HCIVERSION_INIT

Bits	Reset	R/W	Description
15:8	Unknown	R	T_XUSB_XHCI_CAP_REG0_RSVD: 0h: RSVD_00
7:0	Unknown	R	T_XUSB_XHCI_CAP_REG0_CAPLENGTH: 20h: CAPLENGTH_INIT

23.9.12.2 T_XUSB_XHCI_CAP_HCSPARAMS1

Offset: 0x04 | Read/Write: R

Bits	Reset	R/W	Description
31:24	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS1_MAXPORTS: 10h: MAXPORTS_INIT
23	0	R	Reserved
22:19	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS1_RSVD0: 0h: RSVD0_00
18:8	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS1_MAXINTRS: 1h: MAXINTRS_INIT
7:0	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS1_MAXSLOTS: FFh: MAXSLOTS_INIT

23.9.12.3 T_XUSB_XHCI_CAP_HCSPARAMS2

Offset: 0x08 | Read/Write: R

Bits	Reset	R/W	Description
31:27	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_MAXSPBLO: 0h: MAXSPBLO_INIT
26	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_SPR: 1h: SPR_TRUE 0h: SPR_FALSE
25:21	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_MAXSPBHI: 0h: MAXSPBHI_INIT
20:8	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_RSVD: 0h: RSVD_00
7:4	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_ERST_MAX: Fh: ERST_MAX_INIT
3:0	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS2_IST: 8h: IST_INIT

23.9.12.4 T_XUSB_XHCI_CAP_HCSPARAMS3

Offset: 0x0c | Read/Write: R

Bits	Reset	R/W	Description
31:16	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS3_U2LAT: 40h: U2LAT_INIT
15:8	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS3_RSVD: 0h: RSVD_00
7:0	Unknown	R	T_XUSB_XHCI_CAP_HCSPARAMS3_U1LAT: 2h: U1LAT_INIT

23.9.12.5 T_XUSB_XHCI_CAP_HCCPARAMS

Offset: 0x10 | Read/Write: R

Bits	Reset	R/W	Description
31:16	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_XECP: 180h: XECP_USBLEGSUP 184h: XECP_SUPPROT_USB3 187h: XECP_SUPPROT_USB2
15:12	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_MAXPSASIZE: Fh: MAXPSASIZE_INIT
11:9	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_RSVD: 0h: RSVD_00
8	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_PAE: 1h: PAE_TRUE 0h: PAE_FALSE
7	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_NSS: 1h: NSS_TRUE 0h: NSS_FALSE
6	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_LTC: 1h: LTC_TRUE 0h: LTC_FALSE
5	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_LHRC: 1h: LHRC_TRUE 0h: LHRC_FALSE
4	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_PIND: 1h: PIND_TRUE 0h: PIND_FALSE
3	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_PPC: 1h: PPC_TRUE 0h: PPC_FALSE
2	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_CSZ: 1h: CSZ_64B 0h: CSZ_32B
1	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_BNC: 1h: BNC_TRUE 0h: BNC_FALSE
0	Unknown	R	T_XUSB_XHCI_CAP_HCCPARAMS_AC64: 1h: AC64_TRUE 0h: AC64_FALSE

23.9.12.6 T_XUSB_XHCI_CAP_DBOFF

Offset: 0x14 | Read/Write: R

Bits	Reset	R/W	Description
31:2	Unknown	R	T_XUSB_XHCI_CAP_DBOFF_OFFSET: 300h: OFFSET_INIT
1:0	Unknown	R	T_XUSB_XHCI_CAP_DBOFF_RSVD: 0h: RSVD_00

23.9.12.7 T_XUSB_XHCI_CAP_RTSOFF

Offset: 0x18 | Read/Write: R

Bits	Reset	R/W	Description
31:5	Unknown	R	T_XUSB_XHCI_CAP_RTSOFF_OFFSET: 40h: OFFSET_INIT
4:0	Unknown	R	T_XUSB_XHCI_CAP_RTSOFF_RSVD: 0h: RSVD_00

23.9.12.8 T_XUSB_XHCI_CAP_RSVD0

Offset: 0x1c | Read/Write: R

Bits	Reset	R/W	Description
31:0	Unknown	R	T_XUSB_XHCI_CAP_RSVD0_F: 0h: F_00

23.9.12.9 T_XUSB_XHCI_OP_USBCMD

Offset: 0x20 | Read/Write: R/W

Bits	Reset	R/W	Description
31:12	0	R	T_XUSB_XHCI_OP_USBCMD_RSVD1: 0h: RSVD1_00 (default)
11	0	R/W	T_XUSB_XHCI_OP_USBCMD_EU3S: 0h: EU3S_DISABLE (default) 1h: EU3S_ENABLE
10	0	R/W	T_XUSB_XHCI_OP_USBCMD_EWE: 0h: EWE_DISABLE (default) 1h: EWE_ENABLE
9	0	RW1C	T_XUSB_XHCI_OP_USBCMD_CRS: 0h: CRS_INIT (default) 1h: CRS_START 0h: CRS_NOOP
8	0	RW1C	T_XUSB_XHCI_OP_USBCMD_CSS: 0h: CSS_INIT (default) 1h: CSS_START 0h: CSS_NOOP
7	0	R/W	T_XUSB_XHCI_OP_USBCMD_LHCRST: 0h: LHCRST_NOT_PENDING (default) 1h: LHCRST_PENDING 1h: LHCRST_SET
6:4	0	R	T_XUSB_XHCI_OP_USBCMD_RSVD0: 0h: RSVD0_00 (default)
3	0	R/W	T_XUSB_XHCI_OP_USBCMD_HSEE: 0h: HSEE_DISABLE (default) 1h: HSEE_ENABLE
2	0	R/W	T_XUSB_XHCI_OP_USBCMD_INTE: 0h: INTE_DISABLE (default) 1h: INTE_ENABLE
1	0	R/W	T_XUSB_XHCI_OP_USBCMD_HCRST: 0h: HCRST_NOT_PENDING (default) 1h: HCRST_PENDING 1h: HCRST_SET

Bits	Reset	R/W	Description
0	0	R/W	T_XUSB_XHCI_OP_USBCMD_RS: 0h: RS_STOP (default) 1h: RS_RUN

23.9.12.10 T_XUSB_XHCI_OP_USBSTS

Offset: 0x24 | Read/Write: R/W

Bits	Reset	R/W	Description
31:13	0	R	T_XUSB_XHCI_OP_USBSTS_RSVD2: 0h: RSVD2_00 (default)
12	0	R	T_XUSB_XHCI_OP_USBSTS_HCE: 0h: HCE_NO_ERROR (default) 1h: HCE_ERROR
11	1	R	T_XUSB_XHCI_OP_USBSTS_CNR: 1h: CNR_NOT_READY (default) 0h: CNR_READY
10	0	RW1C	T_XUSB_XHCI_OP_USBSTS_SRE: 0h: SRE_NOT_PENDING (default) 1h: SRE_PENDING 1h: SRE_CLEAR
9	0	R	T_XUSB_XHCI_OP_USBSTS_RSS: 0h: RSS_NOT_PENDING (default) 1h: RSS_PENDING
8	0	R	T_XUSB_XHCI_OP_USBSTS_SSS: 0h: SSS_NOT_PENDING (default) 1h: SSS_PENDING
7:5	0	R	T_XUSB_XHCI_OP_USBSTS_RSVD1: 0h: RSVD1_00 (default)
4	0	RW1C	T_XUSB_XHCI_OP_USBSTS_PCD: 0h: PCD_NOT_PENDING (default) 1h: PCD_PENDING 1h: PCD_CLEAR
3	0	RW1C	T_XUSB_XHCI_OP_USBSTS_EINT: 0h: EINT_NOT_PENDING (default) 1h: EINT_PENDING 1h: EINT_CLEAR
2	0	RW1C	T_XUSB_XHCI_OP_USBSTS_HSE: 0h: HSE_NOT_PENDING (default) 1h: HSE_PENDING 1h: HSE_CLEAR
1	0	R	T_XUSB_XHCI_OP_USBSTS_RSVD0: 0h: RSVD0_0 (default)
0	1	R	T_XUSB_XHCI_OP_USBSTS_HCH: 1h: HCH_HALTED (default) 0h: HCH_RUNNING

23.9.12.11 T_XUSB_XHCI_OP_PGSZ

Offset: 0x28 | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_OP_PGSZ_RSVD0: 0h: RSVD0_00 (default)
15:0	1	R	T_XUSB_XHCI_OP_PGSZ_PAGESIZE: 1h: PAGESIZE_4K (default)

23.9.12.12 T_XUSB_XHCI_OP_DNCTRL

Offset: 0x34 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	Reserved
15	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N15: 0h: N15_DISABLE (default) 1h: N15_ENABLE
14	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N14: 0h: N14_DISABLE (default) 1h: N14_ENABLE
13	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N13: 0h: N13_DISABLE (default) 1h: N13_ENABLE
12	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N12: 0h: N12_DISABLE (default) 1h: N12_ENABLE
11	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N11: 0h: N11_DISABLE (default) 1h: N11_ENABLE
10	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N10: 0h: N10_DISABLE (default) 1h: N10_ENABLE
9	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N9: 0h: N9_DISABLE (default) 1h: N9_ENABLE
8	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N8: 0h: N8_DISABLE (default) 1h: N8_ENABLE
7	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N7: 0h: N7_DISABLE (default) 1h: N7_ENABLE
6	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N6: 0h: N6_DISABLE (default) 1h: N6_ENABLE
5	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N5: 0h: N5_DISABLE (default) 1h: N5_ENABLE
4	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N4: 0h: N4_DISABLE (default) 1h: N4_ENABLE
3	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N3: 0h: N3_DISABLE (default) 1h: N3_ENABLE

Bits	Reset	R/W	Description
2	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N2: 0h: N2_DISABLE (default) 1h: N2_ENABLE
1	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N1: 0h: N1_DISABLE (default) 1h: N1_ENABLE
0	0	R/W	T_XUSB_XHCI_OP_DNCTRL_N0: 0h: N0_DISABLE (default) 1h: N0_ENABLE

23.9.12.13 T_XUSB_XHCI_OP_CRCR0

Offset: 0x38 | Read/Write: R/W

Bits	Reset	R/W	Description
31:6	0	W	T_XUSB_XHCI_OP_CRCR0_CRPLO: 0h: CRPLO_INIT (default)
5:4	0	R	T_XUSB_XHCI_OP_CRCR0_RSVD0: 0h: RSVD0_00 (default)
3	0	R	T_XUSB_XHCI_OP_CRCR0_CRR: 0h: CRR_STOPPED (default) 1h: CRR_RUNNING
2	0	RW1C	T_XUSB_XHCI_OP_CRCR0_CA: 0h: CA_INIT (default) 1h: CA_ABORT
1	0	RW1C	T_XUSB_XHCI_OP_CRCR0_CS: 0h: CS_INIT (default) 1h: CS_STOP
0	0	RW1C	T_XUSB_XHCI_OP_CRCR0_RCS: 0h: RCS_0 (default) 1h: RCS_1

23.9.12.14 T_XUSB_XHCI_OP_CRCR1

Offset: 0x3c | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	W	T_XUSB_XHCI_OP_CRCR1_CRPHI: 0h: CRPHI_INIT (default)

23.9.12.15 T_XUSB_XHCI_OP_DCBAAP0

Offset: 0x50 | Read/Write: R/W

Bits	Reset	R/W	Description
31:6	0	R/W	T_XUSB_XHCI_OP_DCBAAP0_DCBAAPLO: 0h: DCBAAPLO_INIT (default)
5:0	0	R	T_XUSB_XHCI_OP_DCBAAP0_RSVD0: 0h: RSVD0_00 (default)

23.9.12.16 T_XUSB_XHCI_OP_DCBAAP1

Offset: 0x54 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_OP_DCBAAP1_DCBAAPHI: 0h: DCBAAPHI_INIT (default)

23.9.12.17 T_XUSB_XHCI_OP_CONFIG

Offset: 0x58 | Read/Write: R/W

Bits	Reset	R/W	Description
31:8	0	R	T_XUSB_XHCI_OP_CONFIG_RSVD0: 0h: RSVD0_00 (default)
7:0	0	R/W	T_XUSB_XHCI_OP_CONFIG_MAXSLOTSEN: 0h: MAXSLOTSEN_INIT (default)

23.9.12.18 T_XUSB_XHCI_OP_PORTSC

Offset: 0x420 – 0x510 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R	T_XUSB_XHCI_OP_PORTSC_WPR: 0h: WPR_NOT_PENDING (default) 1h: WPR_PENDING 1h: WPR_SET
30	0	R	T_XUSB_XHCI_OP_PORTSC_DR: 0h: DR_FALSE (default) 1h: DR_TRUE
29:28	0	R	T_XUSB_XHCI_OP_PORTSC_RSVD2: 0h: RSVD2_00 (default)
27	0	R/W	T_XUSB_XHCI_OP_PORTSC_WOE: 0h: WOE_DISABLED (default) 1h: WOE_ENABLED
26	0	R/W	T_XUSB_XHCI_OP_PORTSC_WDE: 0h: WDE_DISABLED (default) 1h: WDE_ENABLED
25	0	R/W	T_XUSB_XHCI_OP_PORTSC_WCE: 0h: WCE_DISABLED (default) 1h: WCE_ENABLED
24	0	R	T_XUSB_XHCI_OP_PORTSC_CAS: 0h: CAS_INIT (default)
23	0	RW1C	T_XUSB_XHCI_OP_PORTSC_CEC: 0h: CEC_NOT_PENDING (default) 1h: CEC_PENDING 1h: CEC_CLEAR
22	0	RW1C	T_XUSB_XHCI_OP_PORTSC_PLC: 0h: PLC_NOT_PENDING (default) 1h: PLC_PENDING 1h: PLC_CLEAR
21	0	RW1C	T_XUSB_XHCI_OP_PORTSC_PRC: 0h: PRC_NOT_PENDING (default) 1h: PRC_PENDING 1h: PRC_CLEAR

Bits	Reset	R/W	Description
20	0	RW1C	T_XUSB_XHCI_OP_PORTSC_OCC: 0h: OCC_NOT_PENDING (default) 1h: OCC_PENDING 1h: OCC_CLEAR
19	0	RW1C	T_XUSB_XHCI_OP_PORTSC_WRC: 0h: WRC_NOT_PENDING (default) 1h: WRC_PENDING 1h: WRC_CLEAR
18	0	RW1C	T_XUSB_XHCI_OP_PORTSC_PEC: 0h: PEC_NOT_PENDING (default) 1h: PEC_PENDING 1h: PEC_CLEAR
17	0	RW1C	T_XUSB_XHCI_OP_PORTSC_CSC: 0h: CSC_NOT_PENDING (default) 1h: CSC_PENDING 1h: CSC_CLEAR
16	0	RW1C	T_XUSB_XHCI_OP_PORTSC_LWS: 0h: LWS_DISABLED (default) 1h: LWS_ENABLED
15:14	0	R/W	T_XUSB_XHCI_OP_PORTSC_PIC: 0h: PIC_OFF (default) 1h: PIC_AMBER 2h: PIC_GREEN 3h: PIC_UNDEFINED
13:10	0	R	T_XUSB_XHCI_OP_PORTSC_PSPD: 0h: PSPD_UNDEFINED (default) 1h: PSPD_FS 2h: PSPD_LS 3h: PSPD_HS 4h: PSPD_SS Fh: PSPD_UNKNOWN
9	1	R/W	T_XUSB_XHCI_OP_PORTSC_PP: 0h: PP_OFF 1h: PP_ON (default)
8:5	4h	RW1C	T_XUSB_XHCI_OP_PORTSC_PLS: 0h: PLS_U0 1h: PLS_U1 2h: PLS_U2 3h: PLS_U3 4h: PLS_DISABLED (default) 5h: PLS_RXDETECT 6h: PLS_INACTIVE 7h: PLS_POLLING 8h: PLS_RECOVERY 9h: PLS_HOTRESET Ah: PLS_COMPLIANCE Bh: PLS_LOOPBACK Fh: PLS_RESUME 0h: PLS_GOTO_U0 1h: PLS_GOTO_U1 2h: PLS_GOTO_U2 3h: PLS_GOTO_U3
4	0	R/W	T_XUSB_XHCI_OP_PORTSC_PR: 0h: PR_NOT_PENDING (default) 1h: PR_PENDING 1h: PR_SET
3	0	R	T_XUSB_XHCI_OP_PORTSC_OCA: 0h: OCA_FALSE (default) 1h: OCA_TRUE

Bits	Reset	R/W	Description
2	0	R	T_XUSB_XHCI_OP_PORTSC_RSVD0: 0h: RSVD0_0 (default)
1	0	RW1C	T_XUSB_XHCI_OP_PORTSC_PED: 0h: PED_DISABLED (default) 1h: PED_ENABLED 1h: PED_CLEAR
0	0	R	T_XUSB_XHCI_OP_PORTSC_CCS: 0h: CCS_NODEV (default) 1h: CCS_DEV

23.9.12.19 T_XUSB_XHCI_OP_PORTPMSCSS

Offset: 0x424 – 0x514 | Read/Write: R/W

Bits	Reset	R/W	Description
31:17	0	R	T_XUSB_XHCI_OP_PORTPMSCSS_RSVD0: 0h: RSVD0_00 (default)
16	0	R/W	T_XUSB_XHCI_OP_PORTPMSCSS_FLA: 0h: FLA_INIT (default)
15:8	0	R/W	T_XUSB_XHCI_OP_PORTPMSCSS_U2TIMEOUT: 0h: U2TIMEOUT_INIT (default)
7:0	0	R/W	T_XUSB_XHCI_OP_PORTPMSCSS_U1TIMEOUT: 0h: U1TIMEOUT_INIT (default)

23.9.12.20 T_XUSB_XHCI_OP_PORTPMSCHS

Offset: 0x424 – 0x514 | Read/Write: R/W

Bits	Reset	R/W	Description
31:28	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_TM: 0h: TM_DISABLE (default) 1h: TM_JSTATE 2h: TM_KSTATE 3h: TM_SE0NAK 4h: TM_PACKET 5h: TM_FORCEEN Fh: TM_ERROR
27:17	0	R	T_XUSB_XHCI_OP_PORTPMSCHS_RSVD0: 0h: RSVD0_00 (default)
16	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_HLE: 0h: HLE_INIT (default)
15:8	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_L1DS: 0h: L1DS_INIT (default)
7:4	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_HIRD: 0h: HIRD_INIT (default)
3	0	R/W	T_XUSB_XHCI_OP_PORTPMSCHS_RWE: 0h: RWE_DISABLED (default) 1h: RWE_ENABLED
2:0	0	R	T_XUSB_XHCI_OP_PORTPMSCHS_L1S: 0h: L1S_INVLD (default) 1h: L1S_SUCCESS 2h: L1S_NYET 3h: L1S_STALL 4h: L1S_ERROR

23.9.12.21 T_XUSB_XHCI_OP_PORTLISC

Offset: 0x428 – 0x458 | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_OP_PORTLISC_RSVD0: 0h: RSVD0_00 (default)
15:0	0	R	T_XUSB_XHCI_OP_PORTLISC_LEC: 0h: LEC_INIT (default)

23.9.12.22 T_XUSB_XHCI_EC_USBLEGSUP

Offset: 0x600 | Read/Write: R/W

Bits	Reset	R/W	Description
31:25	0	R	T_XUSB_XHCI_EC_USBLEGSUP_RSVD1: 0h: RSVD1_00 (default)
24	0	R/W	T_XUSB_XHCI_EC_USBLEGSUP_OSSEM: 0h: OSSEM_INIT (default)
23:17	0	R	T_XUSB_XHCI_EC_USBLEGSUP_RSVD0: 0h: RSVD0_00 (default)
16	0	R/W	T_XUSB_XHCI_EC_USBLEGSUP_BIOSSEM: 0h: BIOSSEM_INIT (default)
15:8	4h	R	T_XUSB_XHCI_EC_USBLEGSUP_NEXT: 4h: NEXT_SUPPROT_USB3 (default)
7:0	1h	R	T_XUSB_XHCI_EC_USBLEGSUP_CAPID: 1h: CAPID_USBLEGSUP (default)

23.9.12.23 T_XUSB_XHCI_EC_USBLEGCTLSTS

Offset: 0x604 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	RW1C	T_XUSB_XHCI_EC_USBLEGCTLSTS_BAR: 0h: BAR_NOT_PENDING (default) 1h: BAR_PENDING 1h: BAR_CLEAR
30	0	RW1C	T_XUSB_XHCI_EC_USBLEGCTLSTS_PCIC: 0h: PCIC_NOT_PENDING (default) 1h: PCIC_PENDING 1h: PCIC_CLEAR
29	0	RW1C	T_XUSB_XHCI_EC_USBLEGCTLSTS_OSOC: 0h: OSOC_NOT_PENDING (default) 1h: OSOC_PENDING 1h: OSOC_CLEAR
28:21	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_RSVD3: 0h: RSVD3_00 (default)
20	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_HSE: 0h: HSE_NOT_PENDING (default) 1h: HSE_PENDING
19:17	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_RSVD2: 0h: RSVD2_00 (default)
16	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_EVI: 0h: EVI_NOT_PENDING (default) 1h: EVI_PENDING

Bits	Reset	R/W	Description
15	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_BAREN: 0h: BAREN_DISABLED (default) 1h: BAREN_ENABLED
14	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_PCIEN: 0h: PCIEN_DISABLED (default) 1h: PCIEN_ENABLED
13	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_OSOEN: 0h: OSOEN_DISABLED (default) 1h: OSOEN_ENABLED
12:5	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_RSVD1: 0h: RSVD1_00 (default)
4	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_HSEEN: 0h: HSEEN_DISABLED (default) 1h: HSEEN_ENABLED
3:1	0	R	T_XUSB_XHCI_EC_USBLEGCTLSTS_RSVD0: 0h: RSVD0_00 (default)
0	0	R/W	T_XUSB_XHCI_EC_USBLEGCTLSTS_SMIEN: 0h: SMIEN_DISABLED (default) 1h: SMIEN_ENABLED

23.9.12.24 T_XUSB_XHCI_EC_SUPPROT_USB3_0

Offset: 0x610 | Read/Write: R

Bits	Reset	R/W	Description
31:24	3h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_0_MAJORREV: 3h: MAJORREV_3 (default)
23:16	0	R	T_XUSB_XHCI_EC_SUPPROT_USB3_0_MINORREV: 0h: MINORREV_0 (default)
15:8	4h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_0_NEXT: 4h: NEXT_SUPPROT_USB2 (default)
7:0	2h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_0_CAPID: 2h: CAPID_SUPPROT_USB3 (default)

23.9.12.25 T_XUSB_XHCI_EC_SUPPROT_USB3_1

Offset: 0x614 | Read/Write: R

Bits	Reset	R/W	Description
31:0	20425355h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_1_NAMESTR: 20425355h: NAMESTR_USB (default)

23.9.12.26 T_XUSB_XHCI_EC_SUPPROT_USB3_2

Offset: 0x618 | Read/Write: R

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_EC_SUPPROT_USB3_2_RSVD0: 0h: RSVD0_00 (default)
15:8	Unknown	R	T_XUSB_XHCI_EC_SUPPROT_USB3_2_PORTCNT:
7:0	1h	R	T_XUSB_XHCI_EC_SUPPROT_USB3_2_PORTOFS: 1h: PORTOFS_VAL (default)

23.9.12.27 T_XUSB_XHCI_EC_SUPPROT_USB3_3

Offset: 0x61c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_EC_SUPPROT_USB3_3_SLOTTYPE: 0h: SLOTTYPE_VAL (default)

23.9.12.28 T_XUSB_XHCI_EC_SUPPROT_USB2_0

Offset: 0x620 | Read/Write: R

Bits	Reset	R/W	Description
31:24	2h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_0_MAJORREV: 2h: MAJORREV_3 (default)
23:16	0h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_0_MINORREV: 0h: MINORREV_0 (default)
15:8	4h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_0_NEXT: 4h: NEXT_DBCAP (default)
7:0	2h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_0_CAPID: 2h: CAPID_SUPPROT_USB3 (default)

23.9.12.29 T_XUSB_XHCI_EC_SUPPROT_USB2_1

Offset: 0x624 | Read/Write: R

Bits	Reset	R/W	Description
31:0	20425355h	R	T_XUSB_XHCI_EC_SUPPROT_USB2_1_NAMESTR: 20425355h: NAMESTR_USB (default)

23.9.12.30 T_XUSB_XHCI_EC_SUPPROT_USB2_2

Offset: 0x628 | Read/Write: R

Bits	Reset	R/W	Description
31:20	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_RSVD1: 0h: RSVD1_00 (default)
19	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_HLC: 0h: HLC_TRUE (default)
18	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_IHI: 0h: IHI_TRUE (default)
17	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_HSO: 0h: HSO_TRUE (default)
16	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_RSVD0: 0h: RSVD0_00 (default)
15:8	Unknown	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_PORTCNT:
7:0	Unknown	R	T_XUSB_XHCI_EC_SUPPROT_USB2_2_PORTOFS:

23.9.12.31 T_XUSB_XHCI_EC_SUPPROT_USB2_3

Offset: 0x62c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_EC_SUPPROT_USB2_3_SLOTTYPE: 0h: SLOTTYPE_VAL (default)

23.9.12.32 T_XUSB_XHCI_EC_DBCAP_DCID

Offset: 0x630 | Read/Write: R

Bits	Reset	R/W	Description
31:21	0	R	Reserved
20:16	1h	R	T_XUSB_XHCI_EC_DBCAP_DCID_DCERSTM: 1h: DCERSTM_VALUE (default)
15:8	0	R	T_XUSB_XHCI_EC_DBCAP_DCID_NEXT: 0h: NEXT_NONE (default)
7:0	Ah	R	T_XUSB_XHCI_EC_DBCAP_DCID_CAPID: Ah: CAPID_DBCAP (default)

23.9.12.33 T_XUSB_XHCI_EC_DBCAP_DCDB

Offset: 0x634 | Read/Write: W

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_EC_DBCAP_DCDB_RSVD1: 0h: RSVD1_00 (default)
15:8	0	W	T_XUSB_XHCI_EC_DBCAP_DCDB_DBTARGET: 0h: DBTARGET_INIT (default)
7:0	0	R	T_XUSB_XHCI_EC_DBCAP_DCDB_RSVD0: 0h: RSVD0_00 (default)

23.9.12.34 T_XUSB_XHCI_EC_DBCAP_DCERSTSZ

Offset: 0x638 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_EC_DBCAP_DCERSTSZ_RSVD0: 0h: RSVD0_00 (default)
15:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERSTSZ_ERSTSZ: 0h: ERSTSZ_INIT (default)

23.9.12.35 T_XUSB_XHCI_EC_DBCAP_RSVD0

Offset: 0x63c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_EC_DBCAP_RSVD0_RSVD0: 0h: RSVD0_00 (default)

23.9.12.36 T_XUSB_XHCI_EC_DBCAP_DCERSTBALO

Offset: 0x640 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERSTBALO_ADDRLO: 0h: ADDRLO_INIT (default)
3:0	0	R	T_XUSB_XHCI_EC_DBCAP_DCERSTBALO_RSVD0: 0h: RSVD0_00 (default)

23.9.12.37 T_XUSB_XHCI_EC_DBCAP_DCERSTBAHI

Offset: 0x644 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERSTBAHI_ADDRHI: 0h: ADDRHI_INIT (default)

23.9.12.38 T_XUSB_XHCI_EC_DBCAP_DCERDPLO

Offset: 0x648 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERDPLO_ADDRLO: 0h: ADDRLO_INIT (default)
3	0	R	T_XUSB_XHCI_EC_DBCAP_DCERDPLO_RSVD0: 0h: RSVD0_00 (default)
2:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERDPLO_DESI: 0h: DESI_INIT (default)

23.9.12.39 T_XUSB_XHCI_EC_DBCAP_DCERDPHI

Offset: 0x64c | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCERDPHI_ADDRHI: 0h: ADDRHI_INIT (default)

23.9.12.40 T_XUSB_XHCI_EC_DBCAP_DCCTRL

Offset: 0x650 | Read/Write: R/W

Bits	Reset	R/W	Description
31	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCCTRL_DCE: 0h DCE_DIS (default) 1h DCE_EN
30:24	0	R	T_XUSB_XHCI_EC_DBCAP_DCCTRL_DEVADR: 0h: DEVADR_INIT (default)
23:16	15h	R	T_XUSB_XHCI_EC_DBCAP_DCCTRL_MAXBURST: 15h: MAXBURST_INIT (default)
15:5	0	R	T_XUSB_XHCI_EC_DBCAP_DCCTRL_RSVD0: 0h: RSVD0_00 (default)
4	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCCTRL_DRC: 0h: DRC_INIT (default) 1h: DRC_SET 1h: DRC_CLEAR
3	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCCTRL_HIT: 0h: HIT_FALSE (default) 1h: HIT_TRUE
2	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCCTRL_HOT: 0h: HOT_FALSE (default) 1h: HOT_TRUE
1	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCCTRL_LSE: 0h: LSE_DIS (default) 1h: LSE_EN

Bits	Reset	R/W	Description
0	0	R	T_XUSB_XHCI_EC_DBCAP_DCCTRL_DCR: 0h: DCR_STOP (default) 1h: DCR_RUN

23.9.12.41 T_XUSB_XHCI_EC_DBCAP_DCST

Offset: 0x654 | Read/Write: R

Bits	Reset	R/W	Description
31:24	0	R	T_XUSB_XHCI_EC_DBCAP_DCST_DPN: 0h: DPN_INIT (default)
23:1	0	R	T_XUSB_XHCI_EC_DBCAP_DCST_RSVD: 0h: RSVD_00 (default)
0	0	R	T_XUSB_XHCI_EC_DBCAP_DCST_ER: 0h: ER_EMPTY (default) 1h: ER_NOTEMPTY

23.9.12.42 T_XUSB_XHCI_EC_DBCAP_DCPORTSC

Offset: 0x658 | Read/Write: R/W

Bits	Reset	R/W	Description
31:24	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD4: 0h: RSVD4_00 (default)
23	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_CEC: 0h: CEC_NOT_PENDING (default) 1h: CEC_PENDING 1h: CEC_CLEAR
22	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PLC: 0h: PLC_NOT_PENDING (default) 1h: PLC_PENDING 1h: PLC_CLEAR
21	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PRC: 0h: PRC_NOT_PENDING (default) 1h: PRC_PENDING 1h: PRC_CLEAR
20:18	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD3: 0h: RSVD3_00 (default)
17	0	RW1C	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_CSC: 0h: CSC_NOT_PENDING (default) 1h: CSC_PENDING 1h: CSC_CLEAR
16:14	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD2: 0h: RSVD2_00 (default)
13:10	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PS: 0h: PS_UNDEFINED (default) 4h: PS_SS
9	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD1: 0h: RSVD1_00 (default)

Bits	Reset	R/W	Description
8:5	4h	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PLS: 0h: PLS_U0 1h: PLS_U1 2h: PLS_U2 3h: PLS_U3 4h: PLS_DISABLED (default) 5h: PLS_RXDETECT 6h: PLS_INACTIVE 7h: PLS_POLLING 8h: PLS_RECOVERY 9h: PLS_HOTRESET Ah: PLS_COMPLIANCE Bh: PLS_LOOPBACK Fh: PLS_RESUME
4	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PR: 0h: PR_NORST (default) 1h: PR_RST
3:2	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_RSVD0: 0h: RSVD0_00 (default)
1	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_PED: 0h: PED_DIS (default) 1h: PED_EN
0	0	R	T_XUSB_XHCI_EC_DBCAP_DCPORTSC_CCS: 0h: CCS_NOCON (default) 1h: CCS_CON

23.9.12.43 T_XUSB_XHCI_EC_DBCAP_RSVD1

Offset: 0x65c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_EC_DBCAP_RSVD1_RSVD0: 0h: RSVD0_00 (default)

23.9.12.44 T_XUSB_XHCI_EC_DBCAP_DCECPLO

Offset: 0x660 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCECPLO_ADDRLO: 0h: ADDRLO_INIT (default)
3:0	0	R	T_XUSB_XHCI_EC_DBCAP_DCECPLO_RSVD0: 0h: RSVD0_00 (default)

23.9.12.45 T_XUSB_XHCI_EC_DBCAP_DCECPHI

Offset: 0x664 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_DCECPHI_ADDRHI: 0h: ADDRHI_INIT (default)

23.9.12.46 T_XUSB_XHCI_EC_DBCAP_INFO0

Offset: 0x668 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R/W	T_XUSB_XHCI_EC_DBCAP_INFO0_VENDORID: 0h: VENDORID_INIT (default)
15:8	0	R	Reserved
7:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_INFO0_PROTOCOL: 0h: PROTOCOL_VENDOR (default) 1h: PROTOCOL_GNU

23.9.12.47 T_XUSB_XHCI_EC_DBCAP_INFO1

Offset: 0x66c | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R/W	T_XUSB_XHCI_EC_DBCAP_INFO1_DEV_REV: 0h: DEV_REV_INIT (default)
15:0	0	R/W	T_XUSB_XHCI_EC_DBCAP_INFO1_PRODUCTID: 0h: PRODUCTID_INIT (default)

23.9.12.48 T_XUSB_XHCI_RT_MFINDEX

Offset: 0x800 | Read/Write: R

Bits	Reset	R/W	Description
31:14	0	R	T_XUSB_XHCI_RT_MFINDEX_RSVD0: 0h: RSVD0_00
13:0	Unknown	R	T_XUSB_XHCI_RT_MFINDEX_MFINDEX:

23.9.12.49 T_XUSB_XHCI_RT_IMAN

Offset: 0x820 | Read/Write: R/W

Bits	Reset	R/W	Description
31:2	0	R	T_XUSB_XHCI_RT_IMAN_RSVD0: 0h: RSVD0_00
1	0	R/W	T_XUSB_XHCI_RT_IMAN_IE: 0h: IE_DISABLED (default) 1h: IE_ENABLED
0	0	RW1C	T_XUSB_XHCI_RT_IMAN_IP: 0h: IP_NOT_PENDING (default) 1h: IP_PENDING 1h: IP_CLEAR

23.9.12.50 T_XUSB_XHCI_RT_IMOD

Offset: 0x824 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	Unknown	R/W	T_XUSB_XHCI_RT_IMOD_IMODC:
15:0	0FA0h	R/W	T_XUSB_XHCI_RT_IMOD_IMODI: FA0h: IMODI_INIT (default)

23.9.12.51 T_XUSB_XHCI_RT_ERSTSZ

Offset: 0x828 | Read/Write: R/W

Bits	Reset	R/W	Description
31:16	0	R	T_XUSB_XHCI_RT_ERSTSZ_RSVD0: 0h: RSVD0_00
15:0	0	R/W	T_XUSB_XHCI_RT_ERSTSZ_SZ: 0h: SZ_INIT (default)

23.9.12.52 T_XUSB_XHCI_RT_ERRSVD

Offset: 0x82c | Read/Write: R

Bits	Reset	R/W	Description
31:0	0	R	T_XUSB_XHCI_RT_ERRSVD_RSVD0: 0h: RSVD0_00

23.9.12.53 T_XUSB_XHCI_RT_ERSTBA0

Offset: 0x830 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_RT_ERSTBA0_ERSTBLO: 0h: ERSTBLO_INIT (default)
3:0	0	R	T_XUSB_XHCI_RT_ERSTBA0_RSVD0: 0h: RSVD0_00

23.9.12.54 T_XUSB_XHCI_RT_ERSTBA1

Offset: 0x834 | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_RT_ERSTBA1_ERSTBHI: 0h: ERSTBHI_INIT (default)

23.9.12.55 T_XUSB_XHCI_RT_ERDP0

Offset: 0x838 | Read/Write: R/W

Bits	Reset	R/W	Description
31:4	0	R/W	T_XUSB_XHCI_RT_ERDP0_DQPTRLO: 0h: DQPTRLO_INIT (default)
2:0	0	R/W	T_XUSB_XHCI_RT_ERDP0_DESI: 0h: DESI_INIT (default)

23.9.12.56 T_XUSB_XHCI_RT_ERDP1

Offset: 0x83c | Read/Write: R/W

Bits	Reset	R/W	Description
31:0	0	R/W	T_XUSB_XHCI_RT_ERDP1_DQPTRHI: 0h: DQPTRHI_INIT (default)

23.9.12.57 T_XUSB_XHCI_DB

Offset: 0xc00 – 0xffc | Read/Write: W

Bits	Reset	R/W	Description
31:16	0	W	T_XUSB_XHCI_DB_STREAMID: 0h: STREAMID_INIT (default)
15:8	0	R	T_XUSB_XHCI_DB_RSVD0: 0h: RSVD0_00
7:0	0	W	T_XUSB_XHCI_DB_TARGET: 0h: TARGET_INIT (default)

23.9.13 USB3 UTMIP Configuration Registers

Note: Current UTMIP configuration registers use settings that support NV software drivers and should NOT be modified. If designing custom drivers, consult your NV representative prior to modifying these registers

23.9.13.1 USB3_UTMIP_PLL_CFG0_0

USB_PHY PLL Configuration Register 0

Note: This register has been moved to the Clock and Reset section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT!

This register was used to configure the PLL inside UTMIP block prior to Tegra 4 devices. This has been defeatured from UTMIP space and moved to Clock and Reset (CAR) space. Refer to the UTMIP_PLL register descriptions in the Clock and Reset Controller section.

23.9.13.2 USB3_UTMIP_PLL_CFG1_0

UTMIP PLL and PLLU Configuration Register 1

Note: This register has been moved to the Clock and Reset section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT!

This register was used to configure the PLL inside the UTMIP block prior to Tegra 4 devices. This has been defeatured from UTMIP space and moved to CAR space. Refer to the UTMIP_PLL register descriptions in the Clock and Reset Controller section.

23.9.13.3 USB3_UTMIP_XCVR_CFG0_0

UTMIP Transceiver Cell Configuration Register 0

Offset: 0x808 | Read/Write: R/W | Reset: 0b00100000001001010110010100000000

Bit	Reset	Description
31:25	0x10	UTMIP_XCVR_HSSLEW_MSB: Most significant bits of HS_SLEW.
24:22	0x0	UTMIP_XCVR_SETUP_MSB: Most significant bits of SETUP.
21	0x1	UTMIP_XCVR_LSBIAS_SEL: Low speed bias selection method for usb transceiver pad
20	0x0	UTMIP_XCVR_DISCON_METHOD: Disconnect method on the usb transceiver pad
19	0x0	UTMIP_FORCE_PDZI_POWERUP: Force PDZI input into power up.
18	0x1	UTMIP_FORCE_PDZI_POWERDOWN: Force PDZI input into power down. (Overrides FORCE_PDZI_POWERUP.)

Bit	Reset	Description
17	0x0	UTMIP_FORCE_PD2_POWERUP: Force PD2 input into power up.
16	0x1	UTMIP_FORCE_PD2_POWERDOWN: Force PD2 input into power down. (Overrides FORCE_PD2_POWERUP.)
15	0x0	UTMIP_FORCE_PD_POWERUP: Force PD input into power up.
14	0x1	UTMIP_FORCE_PD_POWERDOWN: Force PD input into power down. (Overrides FORCE_PD_POWERUP.)
13	0x1	UTMIP_XCVR_TERMEN: Enable HS termination.
12	0x0	UTMIP_XCVR_HSLOOPBACK: Internal loopback inside XCVR cell. Used for IOBIST.
11:10	0x1	UTMIP_XCVR_LSFSLEW: LS falling slew rate control.
9:8	0x1	UTMIP_XCVR_LSRSLEW: LS rising slew rate control.
7:6	0x0	UTMIP_XCVR_FSSLEW: FS slew rate control.
5:4	0x0	UTMIP_XCVR_HSSLEW: HS slew rate control. The two LSBs.
3:0	0x0	UTMIP_XCVR_SETUP: SETUP[3:0] input of XCVR cell. HS driver output control. 4 LSBs.

23.9.13.4 USB3_UTMIP_BIAS_CFG0_0

UTMIP Bias Cell Configuration Register 0

Offset: 0x80c | Read/Write: R/W | Reset: 0bx00000001100000000000110000000000

Bit	Reset	Description
30	0x0	UTMIP_IDDIG_C_VAL: See IDDIG_C_SEL.
29	0x0	UTMIP_IDDIG_C_SEL: 0: IdDig_c = IdDig_c. 1: IdDig_c = IDDIG_C_VAL.
28	0x0	UTMIP_IDDIG_B_VAL: See IDDIG_B_SEL.
27	0x0	UTMIP_IDDIG_B_SEL: 0: IdDig_b = IdDig_b. 1: IdDig_b = IDDIG_B_VAL.
26	0x0	UTMIP_IDDIG_A_VAL: See IDDIG_A_SEL.
25	0x0	UTMIP_IDDIG_A_SEL: 0: IdDig_a = IdDig_a. 1: IdDig_a = IDDIG_A_VAL.
24	0x0	UTMIP_HSDISCON_LEVEL_MSB: Most significant bit of UTMIP_HSDISCON_LEVEL, bit 2
23	0x1	UTMIP_IDPD_VAL: See IDPD_SEL.
22	0x1	UTMIP_IDPD_SEL: 0: Reserved. Refer to the PMC registers for this feature.
21	0x0	UTMIP_IDDIG_VAL: See IDDIG_SEL.
20	0x0	UTMIP_IDDIG_SEL: 0: IdDig = IdDig. 1: IdDig = IDDIG_VAL.
19	0x0	UTMIP_GPI_VAL: See GPI_SEL.
18	0x0	UTMIP_GPI_SEL: 0: StaticGpi = IdDig. 1: StaticGpi = GPI_VAL.
17:15	0x0	UTMIP_ACTIVE_TERM_OFFSET: Active termination control offset.
14:12	0x0	UTMIP_ACTIVE_PULLUP_OFFSET: Active 1.5K pull-up control offset.
11	0x1	UTMIP_OTGPD: Power down OTG circuit.
10	0x1	UTMIP_BIASPD: Power down bias circuit.
9:8	0x0	UTMIP_VBUS_LEVEL_LEVEL: Vbus detector level.
7:6	0x0	UTMIP_SESS_LEVEL_LEVEL: SessionEnd detector level.
5:4	0x0	UTMIP_HSCHIRP_LEVEL: HS chirp detector level.
3:2	0x0	UTMIP_HSDISCON_LEVEL: HS disconnect detector level.

Bit	Reset	Description
1:0	0x0	UTMIP_HSSQUELCH_LEVEL: HS squelch detector level.

23.9.13.5 USB3_UTMIP_HSRX_CFG0_0

UTMIP High Speed Receive Config 0

Offset: 0x810 | Read/Write: R/W | Reset: 0b10010001011001010011010000000000

Bit	Reset	Description
31:30	0x2	UTMIP_KEEP_PATT_ON_ACTIVE: Keep the stay alive pattern on active
29	0x0	UTMIP_ALLOW_CONSEC_UPDN: Allow consecutive ups and downs on the bits, debug only, set to 0.
28	0x1	UTMIP_REALIGN_ON_NEW_PKT: Realign the inertia counters on a new packet
27:24	0x1	UTMIP_PCOUNT_UPDN_DIV: The number of (edges-1) needed to move the sampling point
23:21	0x3	UTMIP_SQUELCH_EOP_DLY: Limit the delay of the squelch at EOP time
20	0x0	UTMIP_NO_STRIPPING: Do not strip incoming data
19:15	0xa	UTMIP_IDLE_WAIT: Number of idle cycles to declare IDLE.
14:10	0xd	UTMIP_ELASTIC_LIMIT: Depth of elastic input store
9	0x0	UTMIP_ELASTIC_OVERRUN_DISABLE: Do not declare overrun errors until overflow of FIFO
8	0x0	UTMIP_ELASTIC_UNDERRUN_DISABLE: Do not declare underrun errors
7	0x0	UTMIP_PASS_CHIRP: When in Chirp Mode, allow chirp rx data through
6	0x0	UTMIP_PASS_FEEDBACK: Pass through the feedback, do not block it.
5:4	0x0	UTMIP_PCOUNT_INERTIA: Retime the path.
3:2	0x0	UTMIP_PHASE_ADJUST: Based on incoming edges and current sampling position, adjust phase
1	0x0	UTMIP_THREE_SYNCBITS: Sync pattern detection needs 3 consecutive samples instead of 4
0	0x0	UTMIP_USE4SYNC_TRAN: Require 4 sync pattern transitions (01) instead of 3

23.9.13.6 USB3_UTMIP_HSRX_CFG1_0

UTMIP High Speed Receive Config 1

Offset: 0x814 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx010011

Bit	Reset	Description
5:1	0x9	UTMIP_HS_SYNC_START_DLY: How long to wait before start of sync launches RxActive
0	0x1	UTMIP_HS_ALLOW_KEEP_ALIVE: Allow Keep Alive packets

23.9.13.7 USB3_UTMIP_FSLSRX_CFG0_0

UTMIP Full and Low Speed Receive Config 0

Offset: 0x818 | Read/Write: R/W | Reset: 0b11111101010101001000010000101001

Bit	Reset	Description
31	0x1	UTMIP_FSLS_SE1_DRIBBLE_FILTER: One SE1, do not allow dribble
30	0x1	UTMIP_FSLS_SE1_FILTER: Filter SE1
29	0x1	UTMIP_FSLS_SERIAL_SE0_RCV

Bit	Reset	Description
28:26	0x7	UTMIP_FSLS_UPR_DRIBBLE_SIZE: Do not allow <= dribble bits
25:23	0x2	UTMIP_FSLS_LWR_DRIBBLE_SIZE: Do not allow >= dribble bits
22	0x1	UTMIP_FSLS_EOP_ENDS_AT_SE0: Only look for transitioning out of EOP
21:16	0x14	UTMIP_FSLS_KCOUNT_MAX: Number of K bits in question
15	0x1	UTMIP_FSLS_KCOUNT_LIMIT: Limit the number of bit times a K can last
14	0x0	UTMIP_FSLS_ACTIVE_ON_FULL_SYNC: Require a full sync pattern to declare the data received
13:8	0x4	UTMIP_FSLS_IDLE_WAIT_MAX: 4 bits of of SEO should exceed the time limit
7	0x0	UTMIP_FSLS_IDLE_WAIT_LIMIT: Enable the reset of the state machine on extended SE0
6:1	0x14	UTMIP_FSLS_IDLE_COUNT_MAX: 20 bits of idle should end the packet if FsLsIdleCountLimitCfg=1.
0	0x1	UTMIP_FSLS_IDLE_COUNT_LIMIT: Give up on packet if a long sequence of J

23.9.13.8 USB3_UTMIP_FSLSRX_CFG1_0

UTMIP Full and Low Speed Receive Config 1

Offset: 0x81c | Read/Write: R/W | Reset: 0bxxxxx01000100110011101000000000

Bit	Reset	Description
26	0x0	UTMIP_EARLY_LINE_STATE_FILTER: Assumes line state filtering table is inclusive, not exclusive
25:23	0x4	UTMIP_LS_BOUNCE_LENGTH: Number of clock cycle of LS stable
22:17	0x13	UTMIP_LS_EXTRACTION_COUNT: Phase count on which LS bits are extracted
16:11	0xe	UTMIP_LS_EOP_START_COUNT: Number of SEO clock cycles to block bit extraction
10:5	0x20	UTMIP_LS_SE0_COUNT: Only for this number of 60 MHz of SEO and Idle to end packet
4	0x0	UTMIP_LS_LENIENT_DRIBBLE: Allow for large dribble in low speed mode
3	0x0	UTMIP_FS_LENIENT_DRIBBLE: Allow for large dribble in full speed mode
2	0x0	UTMIP_FS_WEAK_SYNC: Only look for a KK pattern, instead of KJKK
1	0x0	UTMIP_FS_DEBOUNCE: Whether full speed uses debouncing
0	0x0	UTMIP_FS_EOP_LENGTH: Whether full speed EOP is determined within 3(0) or 4(1) 60 MHz cycles

23.9.13.9 USB3_UTMIP_TX_CFG0_0

UTMIP Transmit Config Signals

Offset: 0x820 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxx00010000001000000000

Bit	Reset	Description
19	0x0	UTMIP_FS_PREAMBLE_J: output enable sends an initial J before sync pattern
18	0x0	UTMIP_FS_POSTAMBLE_OUTPUT_ENABLE: output enable turns off 1/2 cycle after
17	0x0	UTMIP_FS_PREAMBLE_OUTPUT_ENABLE: output enable turns on 1/2 cycle before
16	0x1	UTMIP_FSLS_ALLOW_SOP_TX_STUFF_ERR: Allow SOP to be source of transmit error stuffing
15	0x0	UTMIP_HS_READY_WAIT_FOR_VALID
14:10	0x0	UTMIP_HS_TX_IPG_DLY
9	0x1	UTMIP_HS_DISCON_EOP_ONLY: Only check during EOP

Bit	Reset	Description
8	0x0	UTMIP_HS_DISCON_DISABLE: Disable high speed disconnect
7	0x0	UTMIP_HS_POSTAMBLE_OUTPUT_ENABLE: output enable turns off 1 cycle after
6	0x0	UTMIP_HS_PREAMBLE_OUTPUT_ENABLE: output enable turns on 1 cycle before
5	0x0	UTMIP_SIE_RESUME_ON_LINESTATE: SIE, not macrocell, detects Line State change to resume
4	0x0	UTMIP_SOF_ON_NO_STUFF: SOF when OpMode 3 -- perhaps, when sending controller made packets
3	0x0	UTMIP_SOF_ON_NO_ENCODE: SOF when OpMode 2 -- not likely, for Chirp
2	0x0	UTMIP_NO_STUFFING: No bit stuffing, static programming
1	0x0	UTMIP_NO_ENCODING: No encoding, static programming
0	0x0	UTMIP_NO_SYNC_NO_EOP: Do not send SYNC or EOP

23.9.13.10 USB3_UTMIP_MISC_CFG0_0

UTMIP Miscellaneous Configurations

Offset: 0x824 | Read/Write: R/W | Reset: 0bx00000111110000000000000001111000

Bit	Reset	Description
30:27	0x0	UTMIP_DPDM_OBSERVE_SEL: Select DP/DM obs signals
26	0x0	UTMIP_DPDM_OBSERVE: Use DP/DM as obs bus
25	0x1	UTMIP_KEEP_XCVR_PD_ON_SOFT_DISCON
24	0x1	UTMIP_ALLOW_LS_ON_SOFT_DISCON
23	0x1	UTMIP_FORCE_FS_DISABLE_ON_DEV_CHIRP
22	0x1	UTMIP_SUSPEND_EXIT_ON_EDGE: Suspend exit requires edge or simply a value...
21	0x1	UTMIP_LS_TO_FS_SKIP_4MS: Do not block changes for 4 ms when going from LS to FS (should not happen)
20:19	0x0	UTMIP_INJECT_ERROR_TYPE: Force error insertion into RX path. (Used for IOBIST.) 0 = DISABLE 1 = BIT_ERR 2 = RX_ERR 3 = BIT_RX_ERR
18	0x0	UTMIP_FORCE_HS_CLOCK_ON: Force HS clock always on.
17	0x0	UTMIP_DISABLE_HS_TERM: Force HS termination inactive.
16	0x0	UTMIP_FORCE_HS_TERM: Force HS termination active.
15	0x0	UTMIP_DISABLE_PULLUP_DP: Force DP pull-up inactive. (Overrides FORCE_PULLUP_DP.)
14	0x0	UTMIP_DISABLE_PULLUP_DM: Force DM pull-up inactive. (Overrides FORCE_PULLUP_DM.)
13	0x0	UTMIP_DISABLE_PULLDN_DP: Force DP pull-down inactive. (Overrides FORCE_PULLDN_DP.)
12	0x0	UTMIP_DISABLE_PULLDN_DM: Force DM pull-down inactive. (Overrides FORCE_PULLDN_DM.)
11	0x0	UTMIP_FORCE_PULLUP_DP: Force DP pull-up active.
10	0x0	UTMIP_FORCE_PULLUP_DM: Force DM pull-up active.
9	0x0	UTMIP_FORCE_PULLDN_DP: Force DP pull-down active.
8	0x0	UTMIP_FORCE_PULLDN_DM: Force DM pull-down active.
7:5	0x3	UTMIP_STABLE_COUNT: Number of cycles of crystal clock of signal not changing to consider stable.

Bit	Reset	Description
4	0x1	UTMIP_STABLE_ALL: Determines if all signal need to be stable to not change a config.
3	0x1	UTMIP_NO_FREE_ON_SUSPEND: Do not use free running terminations during suspend.
2	0x0	UTMIP_NEVER_FREE_RUNNING_TERMS: Ignore free running terminations, even when no clock
1	0x0	UTMIP_ALWAYS_FREE_RUNNING_TERMS: Use free running terminations at all time
0	0x0	UTMIP_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

23.9.13.11 USB3_UTMIP_MISC_CFG1_0

UTMIP Miscellaneous Configurations

Offset: 0x828 | Read/Write: R/W | Reset: 0bx10000000001100110000000000100100

Bit	Reset	Description
30	0x1	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
29	0x0	UTMIP_LINESTATE_BYPASS: Bypass Line State reclocking logic
28	0x0	UTMIP_LINESTATE_NEG: Use negative edge sync for line state
27	0x0	UTMIP_LINESTATE_XCVRSEL3: 0: Use FS filtering on line state when XcvrSel=3 1: Use LS filtering on line state when XcvrSel=3
26:25	0x0	UTMIP_OBS_SEL
24	0x0	UTMIP_FSLS_TDM
23	0x0	UTMIP_FORCE_IOBIST_CLK_ON
22:18	0x6	UTMIP_PLL_ACTIVE_DLY_COUNT: Reserved. Refer to the Clock and Reset registers for this feature.
17:6	0x600	UTMIP_PLLU_STABLE_COUNT: Reserved. Refer to the Clock and Reset registers for this feature.
5	0x1	UTMIP_RX_ERROR_CNT_CLR
4	0x0	UTMIP_RX_ERROR_CNT_EN
3	0x0	UTMIP_FLIP_FSLS_POLARITY
2	0x1	UTMIP_SUSPEND_TERMSEL
1:0	0x0	UTMIP_XCVRSEL3 Bit 0: 0: 0xa5 -> treat as KeepAlive 1: treat as regular packet Bit 1: 0: Turn on FS EOP detection 1: Turn off FS EOP detection

23.9.13.12 USB3_UTMIP_DEBOUNCE_CFG0_0

UTMIP Avalid and Bvalid Debounce

Debounce values IdDig, Avalid, Bvalid, VbusValid, VbusWakeUp, and SessEnd. Each of these signals has its own debouncer, and for each of those, one out of 2 debouncing times can be chosen (BIAS_DEBOUNCE_A or BIAS_DEBOUNCE_B.)

The values of DEBOUNCE_A and DEBOUNCE_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2MHz) / 4

So to program a 1 ms debounce for BIAS_DEBOUNCE_A, BIAS_DEBOUNCE_A[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 0x82c | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_BIAS_DEBOUNCE_B: simulation value -- Used for interrupts
15:0	0xffff	UTMIP_BIAS_DEBOUNCE_A: simulation value -- Used for interrupts

23.9.13.13 USB3_UTMIP_BAT_CHRG_CFG0_0

UTMIP Battery Charger Configuration

Offset: 0x830 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxx000000xx000001

Bit	Reset	Description
13:8	0x0	UTMIP_CHRG_DEBOUNCE_TIMESCALE: Debouncer time scaling, factor-1 to slow down debouncing by. So 0 is 1, 1 is 2, etc.
5	0x0	UTMIP_OP_I_SRC_EN
4	0x0	UTMIP_ON_SRC_EN
3	0x0	UTMIP_OP_SRC_EN
2	0x0	UTMIP_ON_SINK_EN
1	0x0	UTMIP_OP_SINK_EN
0	0x1	UTMIP_PD_CHRG: Power down charger circuit

23.9.13.14 USB3_UTMIP_SPARE_CFG0_0

UTMIP Spare Configuration Bits

Offset: 0x834 | Read/Write: R/W | Reset: 0b11111111111111110000000011111000

Bit	Reset	Description
31:0	-65288	UTMIP_SPARE: Spare register bits 0: HS_RX_IPG_ERROR_ENABLE 1: HS_RX_FLUSH_ALAP. Flush as late as possible 2: HS_RX_LATE_SQUELCH. Delay Squelch by 1 CLK480 cycle. 31 to 3: Reserved

23.9.13.15 USB3_UTMIP_XCVR_CFG1_0

UTMIP Transceiver Cell Configuration Register 1

Offset: 0x838 | Read/Write: R/W | Reset: 0bxxxx00000001100010000001000010101

Bit	Reset	Description
27:26	0x0	UTMIP_XCVR_RPU_RANGE_ADJ: 1.5k pull-up resistor range shift
25:24	0x0	UTMIP_XCVR_HS_IREF_CAP: High Speed Iref cap control for bias current stability
23:22	0x0	UTMIP_XCVR_SPARE: Spare bits for USB transceiver pad.
21:18	0x6	UTMIP_XCVR_TERM_RANGE_ADJ: Range adjustment on terminations.
17	0x0	UTMIP_RCTRL_SW_SET: Use a software override on RCTRL instead of automatic bias control
16:12	0x8	UTMIP_RCTRL_SW_VAL: Encoded value to use on RCTRL when software override is enabled, 0 to 16 only
11	0x0	UTMIP_TCTRL_SW_SET: Use a software override on TCTRL instead of automatic bias control
10:6	0x8	UTMIP_TCTRL_SW_VAL: Encoded value to use on TCTRL when software override is enabled, 0 to 16 only
5	0x0	UTMIP_FORCE_PDDR_POWERUP: Force PDDR input into power up.
4	0x1	UTMIP_FORCE_PDDR_POWERDOWN: Force PDDR input into power down. (Overrides FORCE_PDDR_POWERUP.)
3	0x0	UTMIP_FORCE_PDCHRP_POWERUP: Force PDCHRP input into power up.
2	0x1	UTMIP_FORCE_PDCHRP_POWERDOWN: Force PDCHRP input into power down. (Overrides FORCE_PDCHRP_POWERUP.)
1	0x0	UTMIP_FORCE_PDDISC_POWERUP: Force PDDISC input into power up.
0	0x1	UTMIP_FORCE_PDDISC_POWERDOWN: Force PDDISC input into power down. (Overrides FORCE_PDDISC_POWERUP.)

23.9.13.16 USB3_UTMIP_BIAS_CFG1_0

UTMIP Bias Cell Configuration Register 1

Offset: 0x83c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxx00000000101101

Bit	Reset	Description
13:8	0x0	UTMIP_BIAS_DEBOUNCE_TIMESCALE: Debouncer time scaling, factor - 1 to slow down debouncing by. So 0 is 1, 1 is 2, etc.
7:3	0x5	UTMIP_BIAS_PDTRK_COUNT: Control the BIAS cell power down lag. The lag should be 20 µs. For a crystal clock of 13 MHz, it should be set to 5.
2	0x1	UTMIP_VBUS_WAKEUP_POWERDOWN: Reserved. Refer to the PMC registers for this feature.
1	0x0	UTMIP_FORCE_PDTRK_POWERUP: Force PDTRK input into power up.
0	0x1	UTMIP_FORCE_PDTRK_POWERDOWN: Force PDTRK input into power down. (Overrides FORCE_PDTRK_POWERUP.)

23.9.13.17 USB3_UTMIP_BIAS_STS0_0

UTMIP Bias Cell Status Register 0

Offset: 0x840 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	UTMIP_TCTRL: Thermal encoding output from USB bias pad.
15:0	X	UTMIP_RCTRL: Thermal encoding output from USB bias pad.

23.9.13.18 USB3_UTMIP_CHRG_DEB_CFG0_0

UTMIP VDcd_Det and VDat_Det debounce

Debounce values VDcd_Det and VDat_Det. Each of these signals has its own debouncer. For each of those, one out of 2 debouncing times can be chosen (CHRG_DEBOUNCE_PERIOD_A or CHRG_DEBOUNCE_PERIOD_B).

The values of DEBOUNCE_PERIOD_A and DEBOUNCE_PERIOD_B are calculated as follows:

0xffff -> No debouncing at all

ms = *1000 / (1/19.2MHz) / 4

So to program a 1 ms debounce for CHG_DEBOUNCE_PERIOD, we have: CHG_DEBOUNCE_PERIOD[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 0x844 | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	UTMIP_CHRG_DEBOUNCE_PERIOD_B: simulation value -- Used for interrupts
15:0	0xffff	UTMIP_CHRG_DEBOUNCE_PERIOD_A: simulation value -- Used for interrupts

23.9.13.19 USB3_UTMIP_MISC_STS0_0

UTMIP MISC Status Register

Offset: 0x848 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
21	X	UTMIP_FS_DRV_EN: Indicates when the controller is driving on the bus
20:0	X	UTMIP_SPARE_FUSES: Spare Fuses value, to keep the connections preserved

23.9.13.20 USB3_UTMIP_PMC_WAKEUP0_0

UTMIP PMC Wakeup Value

Offset: 0x84c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
1	RO	X	UTMIP_LINE_WAKEUP_EVENT_INT_STS: Indicates the status of the PMC wakeup event
0	RW	0x0	UTMIP_LINE_WAKEUP_EVENT_INT_ENB: Interrupt mask for the PMC wakeup event

23.9.13.21 USB3_ UHSIC_PLL_CFG0_0

UHSIC PHY PLL Configuration Register 0

Offset: 0xc00 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
0	0x0	UHSIC_PLL_SPARE: Reserved

23.9.13.22 USB3_ UHSIC_PLL_CFG1_0

UHSIC PLL and PLLU Configuration Register 1

This register is used to configure the PHY PLL contained in the UHSIC module as well as the PLLU power up and down.

PLL CONFIGURATION and PARAMETERS

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

Crystal clock -> enters PLLU to generate 12 MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock

The following parameters control the bring-up of the PLLs:

Coming out of reset or suspend

PllUOnState: start pllu_enable_count and pll_lock_count

Wait ~1 µs to enable the PLL_U (pllu_enable_count == ClkXtal * PLLU_ENABLE_DLY_COUNT * 8)

Wait ~1 ms until PLLU is stable (pll_lock_count == ClkXtal * PLLU_STABLE_COUNT * 256) => USB_PHY PLL_ENABLE

Numbers for a 19.2 MHz crystal clock

- PLLU_ENABLE_DLY_COUNT[4:0] = 1 * 19.2 / 8 = 2.4 = 3 = 0x03
- PLLU_STABLE_COUNT[11:0] = 1000 * 19.2 / 256 = 75 = 0x4b (Note: currently defaults to 0x600)
- XTAL_FREQ_COUNT[11:0] = 2500 * 19.2 / 256 = 187 = 0xbb

Offset: 0xc04 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx0001100000011000000

Bit	Reset	Description
18:14	0x3	UHSIC_PLLU_ENABLE_DLY_COUNT: 1 µs / (1/19.2MHz) = 19 / 8 = 2.36 = 3
13	0x0	UHSIC_FORCE_PLLU_POWERUP
12	0x0	UHSIC_FORCE_PLLU_POWERDOWN
11:0	0xc0	UHSIC_XTAL_FREQ_COUNT: 2.5ms / (1/19.2MHz) = 48000 / 256 = 187 = 0xBB

23.9.13.23 USB3_ UHSIC_HSRX_CFG0_0

UHSIC High Speed Receive Config 0

Offset: 0xc08 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx0010100111000111000

Bit	Reset	Description
18	0x0	UHSIC_NO_STRIPPING: Do not strip incoming data
17:13	0xa	UHSIC_IDLE_WAIT: Number of idle cycles to declare IDLE.
12:8	0xe	UHSIC_ELASTIC_OVERRUN_LIMIT

Bit	Reset	Description
7	0x0	UHSIC_ELASTIC_OVERRUN_DISABLE: Do not declare overrun errors until overflow of FIFO
6:2	0xe	UHSIC_ELASTIC_UNDERRUN_LIMIT
1	0x0	UHSIC_ELASTIC_UNDERRUN_DISABLE: Do not declare underrun errors
0	0x0	UHSIC_PASS_FEEDBACK: Pass through the feedback, do not block it.

23.9.13.24 USB3_ UHSIC_HSRX_CFG1_0

UHSIC High Speed Receive Config 1

Offset: 0xc0c | Read/Write: R/W | Reset: 0bxxxxxxxx10001000001010010001 0011

Bit	Reset	Description
23:20	0x8	UHSIC_TX_BLOCK_CNT: Controls how long after the end of transmission the receive path is blocked
19:14	0x20	UHSIC_RX_STROBE_DLY_TRIMMER: Number of delays cells between UH_RX_STROBE and RxStrobeClk in zero cycle path
13:9	0x14	UHSIC_INPUT_FIFO_DEPTH: Depth of the 2-bit wide input FIFO. Maximum depth is 20. Can be tuned
8	0x1	UHSIC_LINE_STATE_RESUME_FAKE_SE0: When enabled, send an SE0 for 2 LS symbols at the end of ResumeK
7	0x0	UHSIC_LINE_STATE_BYPASS: Bypass Line State reclocking logic
6	0x0	UHSIC_EARLY_LINE_STATE_FILTER: Assumes line state filtering table is inclusive, not exclusive
5:1	0x9	UHSIC_HS_SYNC_START_DLY: How long to wait before start of sync launches RxActive
0	0x1	UHSIC_HS_ALLOW_KEEP_ALIVE: Allow Keep Alive packets

23.9.13.25 USB3_ UHSIC_TX_CFG0_0

UHSIC Transmit Config Signals

Offset: 0xc10 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx1000000000

Bit	Reset	Description
9	0x1	UHSIC_HS_READY_WAIT_FOR_VALID
8	0x0	UHSIC_PACKET_INVERT_DATA: Invert data during a regular packet
7	0x0	UHSIC_PACKET_FORCE_STROBE_LOW: Force STROBE low during a regular instead of toggling it
6	0x0	UHSIC_HS_POSTAMBLE_OUTPUT_ENABLE: output enable turns off 1 cycle after
5	0x0	UHSIC_SIE_RESUME_ON_LINESTATE: SIE, not macrocell, detects Line State change to resume
4	0x0	UHSIC_SOF_ON_NO_STUFF: SOF when OpMode 3 -- perhaps, when the sending controller made packets
3	0x0	UHSIC_SOF_ON_NO_ENCODE: SOF when OpMode 2 -- not likely, for Chirp
2	0x0	UHSIC_NO_STUFFING: No bit stuffing, static programming
1	0x0	UHSIC_NO_ENCODING: No encoding, static programming
0	0x0	UHSIC_NO_SYNC_NO_EOP: Do not send SYNC or EOP

23.9.13.26 USB3_ UHSIC_MISC_CFG0_0

UHSIC Miscellaneous Configurations

Offset: 0xc14 | Read/Write: R/W | Reset: 0bxxxxxxxxxxx00100010011110100011110

Bit	Reset	Description
20	0x0	UHSIC_DISABLE_BUSRESET: When 1, the PHY will not send out BusReset during XcvrSelect0, TermSelect0, and Opmode2. It will send out a non-bit-stuffed, non-encoded packet instead.
19	0x0	UHSIC_FORCE_TERMSEL: Value to be forced on TermSelect when FORCE_XCVR_MODE is set.
18	0x1	UHSIC_EXTEND_BK_ACTIVE: Drive the bus keeper one cycle longer when going out of IDLE
17:16	0x0	UHSIC_FORCE_XCVRSEL: Value to be forced on XcvrSelect when FORCE_XCVR_MODE is set.
15	0x0	UHSIC_FORCE_XCVR_MODE: 1: Force the values of XcvrSelect and TermSelect via config bits instead of via the controller
14	0x1	UHSIC_SYMMETRIC_CONNECT_DATA 0: DATA goes high before STROBE goes low and low before STROBE goes high. 1: DATA goes high before STROBE goes low and goes low *after* STROBE goes high.
13	0x0	UHSIC_ASYNC_CONNECT_DATA 0: DATA keeps setup and hold requirements during CONNECT. 1: DATA moves together with STROBE
12	0x0	UHSIC_LONG_CONNECT_STROBE 0: STROBE is 2 periods long during connect. 1: STROBE is 3 periods long during connect
11	0x1	UHSIC_ACTIVE_BK_DRIVE_RX: 1: Use RX state (EOP, etc.) to determine starting time to drive bus keeper instead of waiting for IDLE detection.
10	0x1	UHSIC_ACTIVE_BK_DRIVE_TX: 1: Use TX state to determine starting time to drive bus keeper instead of waiting for IDLE detection.
9	0x1	UHSIC_DETECT_SHORT_IDLE 0: Use 3 edges (negative and positive) to detect an idle state on the line. 1: Use 4 edges.
8	0x0	UHSIC_DETECT_SHORT_CONNECT: 0: Use 3 edges (negative and positive) to detect a connect state on the line. 1: Use 4 edges.
7	0x1	UHSIC_SUSPEND_EXIT_ON_EDGE: Suspend exit requires edge or simply a value.
6:5	0x0	UHSIC_INJECT_ERROR_TYPE: Force error insertion into RX path. (Used for IOBIST.) 0 = DISABLE 1 = BIT_ERR 2 = RX_ERR 3 = BIT_RX_ERR
4:2	0x3	UHSIC_STABLE_COUNT: Number of crystal clock cycles of signal not changing to consider stable.
1	0x1	UHSIC_STABLE_ALL: Determines if all signals need to be stable to not change a config.
0	0x0	UHSIC_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

23.9.13.27 USB3_ UHSIC_MISC_CFG1_0

UHSIC Miscellaneous Configurations

Offset: 0xc18 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxx100001100000000010

Bit	Reset	Description
17	0x1	UHSIC_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
16:15	0x0	UHSIC_OBS_SEL: Select which one of 4 observation vectors is presented on the observation bus
14	0x0	UHSIC_FORCE_IOBIST_CLK_ON: Always enable IoBist CLK60. This would be required when you want to use RX_ERROR_CNT_EN.
13:2	0x600	UHSIC_PLLU_STABLE_COUNT: PLLU frequency lock delay.
1	0x1	UHSIC_RX_ERROR_CNT_CLR: Clear IOBST RxError counter
0	0x0	UHSIC_RX_ERROR_CNT_EN: Enable IOBIST RxError counter when not in IOBIST mode. Allows one to read out the number of errors via JTAG during normal operation

23.9.13.28 USB3_ UHSIC_PADS_CFG0_0

UHSIC Pads Settings

Offset: 0xc1c | Read/Write: R/W | Reset: 0b00000000100010001000100010001000

Bit	Reset	Description
31:24	0x0	UHSIC_HSIC_OPT: Spare config bits
23:20	0x8	UHSIC_TX_SLEWN: Output slew rate (fall time) adjustment
19:16	0x8	UHSIC_TX_SLEWP: Output slew rate (rise time) adjustment
15:12	0x8	UHSIC_TX_RTUNEN: Fine-tuned 50 Ohm termination resistor for NMOS driver
11:8	0x8	UHSIC_TX_RTUNEP: Fine-tuned 50 Ohm termination resistor for PMOS driver
7:4	0x8	UHSIC_TX_RTERMN: Output impedance adjustment for NMOS driver
3:0	0x8	UHSIC_TX_RTERMP: Output impedance adjustment for PMOS driver

23.9.13.29 USB3_ UHSIC_PADS_CFG1_0

UHSIC Pads Settings

Offset: 0xc20 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxx0011001111101

Bit	Reset	Description
12	0x0	UHSIC_RPU_STROBE: Enable pull up on IO_STROBE
11	0x0	UHSIC_RPU_DATA: Enable pull up on IO_DATA
10	0x1	UHSIC_RPD_STROBE: Enable pull down on IO_STROBE
9	0x1	UHSIC_RPD_DATA: Enable pull down on IO_DATA
8	0x0	UHSIC_LPBK: Internal digital loopback
7	0x0	UHSIC_RX_SEL: 0: Differential read buffers, 1: Single-ended buffers
6	0x1	UHSIC_PD_ZI: Power down single ended receiver
5	0x1	UHSIC_PD_RX: Power down receiver
4	0x1	UHSIC_PD_TRK: Power down tracking circuit

Bit	Reset	Description
3	0x1	UHSIC_PD_TX: Power down transmitter
2	0x1	UHSIC_PD_BG: Power down band-gap and bias generator
1	0x0	UHSIC_IDDQ: Shut down analog blocks for IDDQ testing
0	0x1	UHSIC_AUTO_RTERM_EN: Enable auto-termination

23.9.13.30 USB3_ UHSIC_CMD_CFG0_0

Determine start-up behavior.

When AUTO_NEGIOTIATE == 0

- Firmware is supposed to take care of things.

HOST

if Opmode1: do not drive anything

else

Initial power up:

- Asynchronously drive 00
- After a few crystal clocks, enable bus keepers

Offset: 0xc24 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000101

Bit	Reset	Description
6	0x0	UHSIC_FORCE_ACTIVATED: Force PHY into activated state without connect handshake (both host and device)
5	0x0	UHSIC_PRETEND_CONNECT_DETECT: While in HOST mode, act as if the input stage has seen a CONNECT pulse from the external PHY
4	0x0	UHSIC_FORCE_RESET: While in HOST mode, force global state machine into RESET state
3	0x0	UHSIC_FORCE_CONNECT: Upon rising value of this bit, force device to send connect. Only useful when AUTO_CONNECT is disabled.
2	0x1	UHSIC_AUTO_CONNECT: As device, automatically send Connect during activation.
1	0x0	UHSIC_FORCE_ACTIVATE: Upon rising value of this bit, instruct state machine to go into activation mode. Only useful when AUTO_ACTIVATE is disabled.
0	0x1	UHSIC_AUTO_ACTIVATE: Upon power up, automatically move to activation mode and start going through connect procedure.

23.9.13.31 USB3_ UHSIC_STAT_CFG0_0

Offset: 0xc28 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
31:16	RO	X	UHSIC_CALIOUT
15:8	RO	X	UHSIC_SPARE_STATUS
2:1	RO	X	UHSIC_BUS_STATE
0	RW	0x0	UHSIC_CONNECT_DETECT

23.9.13.32 USB3_ UHSIC_SPARE_CFG0_0

UTMIP Spare Configurations, Spare Configuration Bits

Offset: 0xc2c | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:0	-65536	UHSIC_SPARE: Bit 0: HS_RX_IPG_ERROR_ENABLE Bit 1: HS_RX_FLUSH_ALAP Bit 2: FORCE_TRIM_ZERO Bits 7 :4: RX_DATA_TRIM[3:0] Bit 11:8: RX_STROBE_TRIM[3:0] Bit 12: FORCE_BK_ON Bit 13: BYPASS_INIT_BLOCK Bit 14: FORCE_SM_IDLE Bit 15: FORCE_BK_OFF

23.9.13.33 USB3_ UHSIC_MISC_STS0_0

UHSIC SPARE Fuse Value

Offset: 0xc30 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
0	X	UHSIC_TX_HS_VLD: Indicates when the controller is driving on the bus

23.9.13.34 USB3_ UHSIC_PMC_WAKEUP0_0

UHSIC PMC Wakeup Value

Offset: 0xc34 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
1	RO	X	UHSIC_LINE_WAKEUP_EVENT_INT_STS: Indicates the status of the PMC wakeup event
0	RW	0x0	UHSIC_LINE_WAKEUP_EVENT_INT_ENB: Interrupt mask for the PMC wakeup event

23.9.13.35 USB2_QH_USB2D_QH_EP_0_OUT_0

USB2D Queue Head for OUT Endpoint 0

Offset: 0x1000 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 0. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 0.

23.9.13.36 USB2_QH_USB2D_QH_EP_0_IN_0

USB2D Queue Head for IN Endpoint 0

Offset: 0x1040 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 0. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 0.

23.9.13.37 USB2_QH_USB2D_QH_EP_1_OUT_0

USB2D Queue Head for OUT Endpoint 1

Offset: 0x1080 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 1. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 1.

23.9.13.38 USB2_QH_USB2D_QH_EP_1_IN_0

USB2D Queue Head for IN Endpoint 1

Offset: 0x10c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 1. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 1.

23.9.13.39 USB2_QH_USB2D_QH_EP_2_OUT_0

USB2D Queue Head for OUT Endpoint 2

Offset: 0x1100 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 2. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 2.

23.9.13.40 USB2_QH_USB2D_QH_EP_2_IN_0

USB2D Queue Head for IN Endpoint 2

Offset: 0x1140 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 2. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 2.

23.9.13.41 USB2_QH_USB2D_QH_EP_3_OUT_0

USB2D Queue Head for OUT Endpoint 3

Offset: 0x1180 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 3. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 3.

23.9.13.42 USB2_QH_USB2D_QH_EP_3_IN_0

USB2D Queue Head for IN Endpoint 3

Offset: 0x11c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 3. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 3.

23.9.13.43 USB2_QH_USB2D_QH_EP_4_OUT_0

USB2D Queue Head for OUT Endpoint 4

Offset: 0x1200 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 4. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 4.

23.9.13.44 USB2_QH_USB2D_QH_EP_4_IN_0

USB2D Queue Head for IN Endpoint 4

Offset: 0x1240 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 4. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 4.

23.9.13.45 USB2_QH_USB2D_QH_EP_5_OUT_0

USB2D Queue Head for OUT Endpoint 5

Offset: 0x1280 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 5. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 5.

23.9.13.46 USB2_QH_USB2D_QH_EP_5_IN_0

USB2D Queue Head for IN Endpoint 5

Offset: 0x12c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 5. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 5.

23.9.13.47 USB2_QH_USB2D_QH_EP_6_OUT_0

USB2D Queue Head for OUT Endpoint 6

Offset: 0x1300 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 6. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 6.

23.9.13.48 USB2_QH_USB2D_QH_EP_6_IN_0

USB2D Queue Head for IN Endpoint 6

Offset: 0x1340 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 6. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 6.

23.9.13.49 USB2_QH_USB2D_QH_EP_7_OUT_0

USB2D Queue Head for OUT Endpoint 7

Offset: 0x1380 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 7. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 7.

23.9.13.50 USB2_QH_USB2D_QH_EP_7_IN_0

USB2D Queue Head for IN Endpoint 7

Offset: 0x13c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 7. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 7.

23.9.13.51 USB2_QH_USB2D_QH_EP_8_OUT_0

USB2D Queue Head for OUT Endpoint 8

Offset: 0x1400 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 8. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 8.

23.9.13.52 USB2_QH_USB2D_QH_EP_8_IN_0

USB2D Queue Head for IN Endpoint 8

Offset: 0x1440 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 8. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 8.

23.9.13.53 USB2_QH_USB2D_QH_EP_9_OUT_0

USB2D Queue Head for OUT Endpoint 9

Offset: 0x1480 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 9. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 9.

23.9.13.54 USB2_QH_USB2D_QH_EP_9_IN_0

USB2D Queue Head for IN Endpoint 9

Offset: 0x14c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 9. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 9.

23.9.13.55 USB2_QH_USB2D_QH_EP_10_OUT_0

USB2D Queue Head for OUT Endpoint 10

Offset: 0x1500 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 10. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 10.

23.9.13.56 USB2_QH_USB2D_QH_EP_10_IN_0

USB2D Queue Head for IN Endpoint 10

Offset: 0x1540 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 10. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 10.

23.9.13.57 USB2_QH_USB2D_QH_EP_11_OUT_0

USB2D Queue Head for OUT Endpoint 11

Offset: 0x1580 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 11. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 11.

23.9.13.58 USB2_QH_USB2D_QH_EP_11_IN_0

USB2D Queue Head for IN Endpoint 11

Offset: 0x15c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 11. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 11.

23.9.13.59 USB2_QH_USB2D_QH_EP_12_OUT_0

USB2D Queue Head for OUT Endpoint 12

Offset: 0x1600 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 12. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 12.

23.9.13.60 USB2_QH_USB2D_QH_EP_12_IN_0

USB2D Queue Head for IN Endpoint 12

Offset: 0x1640 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 12. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 12.

23.9.13.61 USB2_QH_USB2D_QH_EP_13_OUT_0

USB2D Queue Head for OUT Endpoint 13

Offset: 0x1680 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 13. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 13.

23.9.13.62 USB2_QH_USB2D_QH_EP_13_IN_0

USB2D Queue Head for IN Endpoint 13

Offset: 0x16c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 13. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 13.

23.9.13.63 USB2_QH_USB2D_QH_EP_14_OUT_0

USB2D Queue Head for OUT Endpoint 14

Offset: 0x1700 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 14. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 14.

23.9.13.64 USB2_QH_USB2D_QH_EP_14_IN_0

USB2D Queue Head for IN Endpoint 14

Offset: 0x1740 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 14. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 14.

23.9.13.65 USB2_QH_USB2D_QH_EP_15_OUT_0

USB2D Queue Head for OUT Endpoint 15

Offset: 0x1780 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for OUT endpoint 15. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for OUT endpoint 15.

23.9.13.66 USB2_QH_USB2D_QH_EP_15_IN_0

USB2D Queue Head for IN Endpoint 15

Offset: 0x17c0 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	USB2D_QH: Queue Head for IN endpoint 15. This is used to store a local Queue Head data structure for either device mode or host mode. In device mode, it holds the Queue Head for IN endpoint 15.

23.9.14 XUSB FPCI Registers

23.9.14.1 XUSB_CFG_ARU_MAILBOX_CMD_0

Offset: 0xe4 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	INT_EN 0: INT_EN_INIT
30	0x0	DEST_XHCI 0: DEST_XHCI_INIT
29	0x0	DEST_SMI 0: DEST_SMI_INIT
28	0x0	DEST_PME 0: DEST_PME_INIT
27	0x0	DEST_FALCON 0: DEST_FALCON_INIT
26:0	0x0	RSVD 0: RSVD_INIT

23.9.14.2 XUSB_CFG_ARU_RST_0

Offset: 0x42c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000

Bit	Reset	Description
2	0x0	CYA_RESET_AUX 0: CYA_RESET_AUX_INIT
1	0x0	CYA_DMAIDLE 0: CYA_DMAIDLE_WAIT 1: CYA_DMAIDLE_FORCE
0	0x0	CFGRST 0: CFGRST_NOT_PENDING 1: CFGRST_PENDING 1: CFGRST_SET

23.9.15 XUSB CSB Registers

23.9.15.1 XUSB_CSB_MEMPOOL_ILOAD_ATTR_0

L2IMEMOP Static Configuration Register

Offset: 0x0101a00 | Read/Write: R/W | Reset: 0b000xxxxxxxxx00000000000000000000

Bit	R/W	Reset	Description
31	RW	0x0	TC 0: TC_DEFAULT
30	RW	0x0	NS 0: NS_DEFAULT
29	RW	0x0	RO 0: RO_DEFAULT
19:8	RW	0x0	SIZE 0: SIZE_DEFAULT
7:0	RO	0x0	RSVD 0: RSVD_DEFAULT

23.9.15.2 XUSB_CSB_MEMPOOL_ILOAD_BASE_LO_0

L2IMEMOP Static Configuration Register

Offset: 0x0101a04 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	R/W	Reset	Description
31:8	RW	0x0	RSVD 0: RSVD_DEFAULT
7:0	RO	0x0	SRC_ADDR 0: SRC_ADDR_DEFAULT

23.9.15.3 XUSB_CSB_MEMPOOL_ILOAD_BASE_HI_0

L2IMEMOP Static Configuration Register

Offset: 0x0101a08 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
7:0	0x0	SRC_ADDR 0: SRC_ADDR_DEFAULT

23.9.15.4 XUSB_CSB_MEMPOOL_L2IMEMOP_SIZE_0

L2IMEMOP Operational Register

Offset: 0x0101a10 | Read/Write: R/W | Reset: 0b0000000xxxxx00000000000000000000

Bit	R/W	Reset	Description
31:24	RW	0x0	SRC_COUNT 0: SRC_COUNT_DEFAULT
19:8	RW	0x0	SRC_OFFSET 0: SRC_OFFSET_DEFAULT
7:0	RO	0x0	RSVD 0: RSVD_DEFAULT

23.9.15.5 XUSB_CSB_MEMPOOL_L2IMEMOP_TRIG_0

L2IMEMOP Operational Register

L2IMEMop Action encodings

bit[0] : 1=> RESULT, 0=> NO RESULT

bit[1] : 1=> OPTIMIZED, 0=> UNOPTIMIZED

bit[7:4] 0001 => LOAD_LOCKED

0010 => UNLOCK

0100 => INVALIDATE_ALL

0101 => QUERY_INDEX

0110 => QUERY_SPACE

Offset: 0x0101a14 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx000000000000000000

Bit	R/W	Reset	Description
31:24	RW	X	ACTION 0x10: L2IMEM_LOAD_LOCKED 0x11: L2IMEM_LOAD_LOCKED_RESULT 0x20: L2IMEM_UNLOCK 0x21: L2IMEM_UNLOCK_RESULT 0x22: L2IMEM_UNLOCK_OPTIMIZED 0x23: L2IMEM_UNLOCK_OPTIMIZED_RESULT 0x40: L2IMEM_INVALIDATE_ALL 0x41: L2IMEM_INVALIDATE_ALL_RESULT 0x50: L2IMEM_QUERY_INDEX_RESULT 0x60: L2IMEM_QUERY_SPACE_RESULT 0xFF: WAITFENCE_RESULT
17:8	RW	0x0	DEST_INDEX 0: DEST_INDEX_DEFAULT
7:0	RO	0x0	RSVD 0: RSVD_DEFAULT

23.9.15.6 XUSB_CSB_MEMPOOL_APMAP_0

Aperture Programming Register

Offset: 0x 010181c | Read/Write: R/W | Reset: 0b1xxxxxxxxxxx0xxxxxxxxxxxxxx00xxxxx000

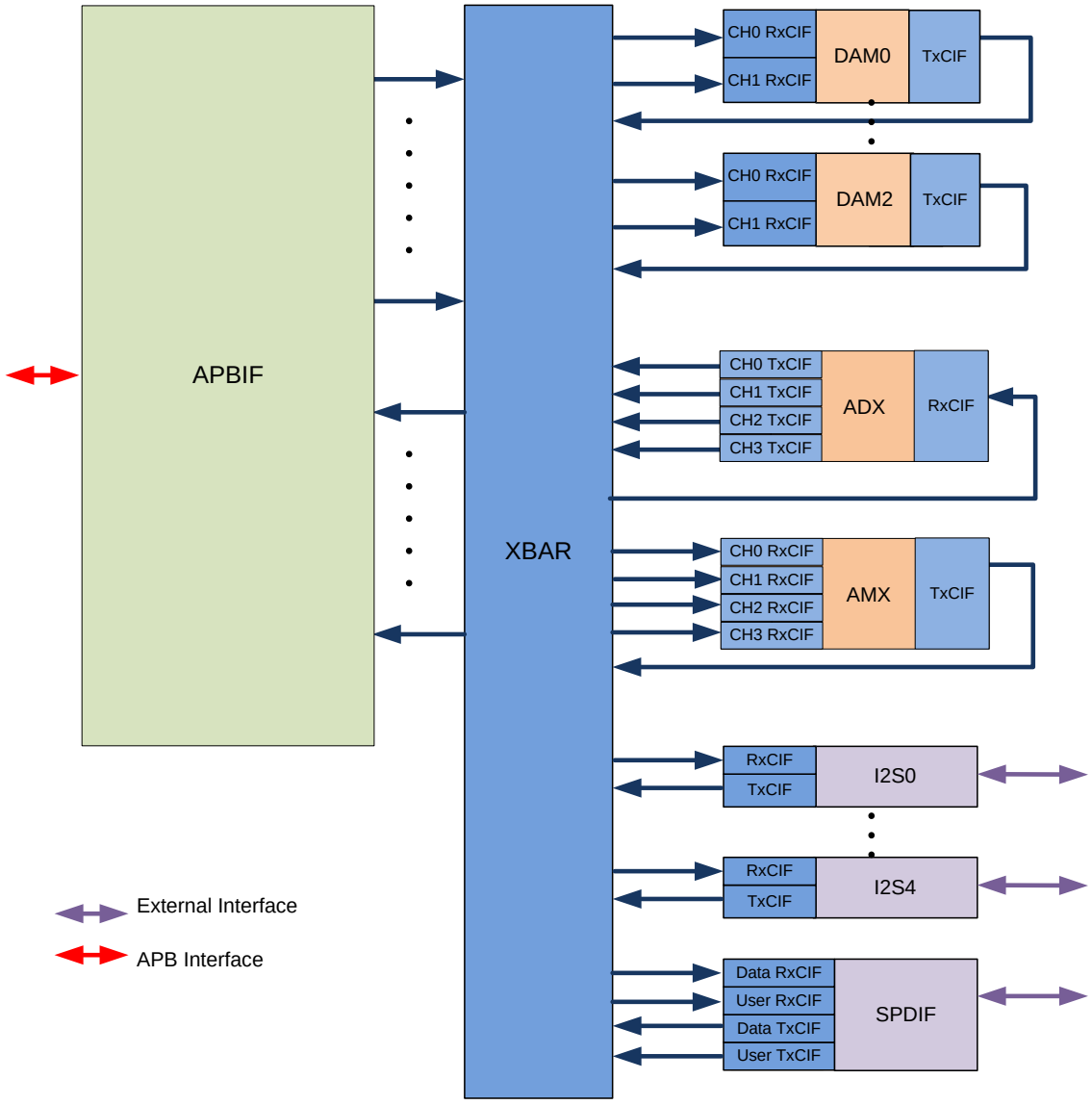
Bit	Reset	Description
31	0x1	BOOTPATH 1: BOOTPATH_DEFAULT
24	0x0	XREQ_READ 0: XREQ_READ_DEFAULT
9:8	0x0	XMAP 0: XMAP_A 1: XMAP_B 2: XMAP_C
2:0	0x0	FDDMA 0: FDDMA_A 1: FDDMA_B 2: FDDMA_C 3: FDDMA_D 4: FDDMA_E

[THIS PAGE INTENTIONALLY LEFT BLANK]

24.0 AUDIO HUB (AHUB)

Tegra® 4 processors support multiple interfaces to the audio devices in the system cellular baseband; different audio codecs; digital audio that is synthesized or decoded by the AVP audio coprocessor or CPU; Bluetooth modules; FM receivers; HDMI audio to a TV or home theater; etc. AHUB can cater to the different interface, protocol, and signal quality requirements of these audio devices.

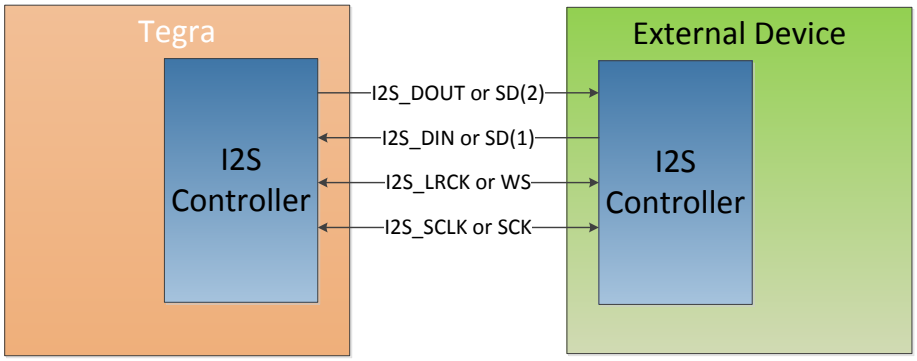
Figure 68: Audio Hub Functional Block Diagram



The Audio Hub (AHUB) consists of

- 5 I^2S controllers and 1 S/PDIF controller to interface with external devices
- A multi-channel DMA unit (APBIF) to interface with memory and internal modules.
- 3 Digital Audio sample-rate conversion and Mixing blocks (DAMs), 1 Audio Multiplexer (AMX), and 1 Audio Demultiplexer (ADX) for audio processing.
- A crossbar switch (XBAR) matrix interconnects all the above modules in any configuration as desired using a CIF interface. CIF interface is an NVIDIA proprietary interface that enables one module in AHUB to send/receive data to/from another module in AHUB.

24.1 I²S Controller



The I²S controller implements full-duplex and bidirectional and single direction point-to-point serial interfaces. It can interface with I2S-compatible products, such as compact disc players, digital audio tape devices, digital sound processors, modems, Bluetooth chips, etc. The I²S controller can operate both as master and slave.

The I²S controller streams synchronous serial audio data between the Tegra 4 device and an external audio device. The controller supports the I²S Left Justified Mode, Right Justified Mode, and DSP mode formats.

The I²S controller also supports the PCM, telephony (network), and Time Division Multiplex (TDM) modes of data transfer. Pulse Code Modulation (PCM) is a standard method used to digitize audio (particularly voice) patterns for transmission over digital communication channels. The Telephony or Network mode is used to transmit and receive data to/from an external mono codec in a slot-based scheme of time-division multiplexing. TDM mode is same as network mode, but in TDM mode, all or any of the slots are active. All of these modes are synced up by a frame-sync signal, much the same as the left-right-channel-clock LRCK in the case of normal I²S communication.

This controller is indirectly connected to the APB bus through the APBIF, and can trigger the APB-bridge DMA to transfer data to and from its FIFOs. Refer to the APBIF section of this document for further details.

Features

- PCM, Network, and TDM mode support
- Can operate in both Master and Slave modes
- Supports I²S, RJM, LJM, and DSP mode data formats
- Supports non- 50:50 mark-space ratio in both Master and Slave modes of I2S, RJM, and LJM
- Contains two FIFOs and control logic. In I2S basic modes, both FIFOs can be configured for Tx or Rx. In PCM/Network mode; FIFO-1 acts as Tx FIFO and FIFO-2 as Rx FIFO
- Buffer memory optimized for peripheral bus utilization, allowing multiple sets of data being stored for both incoming and outgoing data stream per slot (Data-Packed mode in both FIFOs)
- Supports PCM-mode mono and Network mode with independent slot selection for Tx and Rx
- PCM mode supports short and long FSYNC. Long FSYNC can be only 2 clocks wide in Master mode and any number of clocks in slave mode
- Same LRCK/FSYNC for both transmission and reception of data
- Network mode with independent slot selection for both Tx and Rx
- TDM mode with flexibility in number of slots and slot(s) selection
- Capability to drive out a High-Z outside the prescribed slot for transmission
- 6-bit clock divisor to support single- and double-speed for the following rates: 8 KHz, 32 KHz, 44.1 KHz, 48 KHz, 96 KHz, and 128 KHz
- Maximum frequency of the device clock supported is 24 MHz

- The number of slots can be programmed from 1 to a maximum of 8 in TDM mode
- Can transmit and receive word lengths of 8, 16, 24, and 32
- Supports u-Law and A-Law compression/decompression

The I^2S controller supports bidirectional audio streams. It can operate in full-duplex or single directional mode; that is, it can transmit or receive data on both the I2S_DIN and I2S_DOUT pins, based on the register-configuration. The I^2S controller supports input word lengths of 16, 20, 24, or 32 bits. In TDM mode, the word length can be supported in multiples of 4, starting from 8 bits.

In Master mode, the I^2S controller signals the synchronization of all I^2S data transactions. In Master mode, the I^2S controller module generates the SCLK bit clock (I2S_SCLK) and Left-Right channel clock LRCK (I2S_LRC) to the A/D or D/A converter. The A/D or D/A converter synchronizes the input or output audio data based on these clock signals.

In Slave mode, the external A/D or D/A converter provides the synchronization clocks, bit clock (I2S_SCLK) and Left-Right channel clock (I2S_LRCK) to the I^2S controller. The I^2S controller synchronizes the input/output audio data to these clocks.

24.1.1 External Signaling

The I2S interface consists of 4 signals--FSYNC, BITCLK, SDATA1 and SDATA2-- all defined as bidirectional. SDATA1 and SDATA2 associate themselves to FIFO-1 and FIFO-2, respectively. In I2S mode, the SDATA1 and SDATA2 can be configured to be either input or output based on the configuration of the corresponding FIFO.

24.1.1.1 Data Formats in Basic I2S Mode

Figure 69: I2S Mode

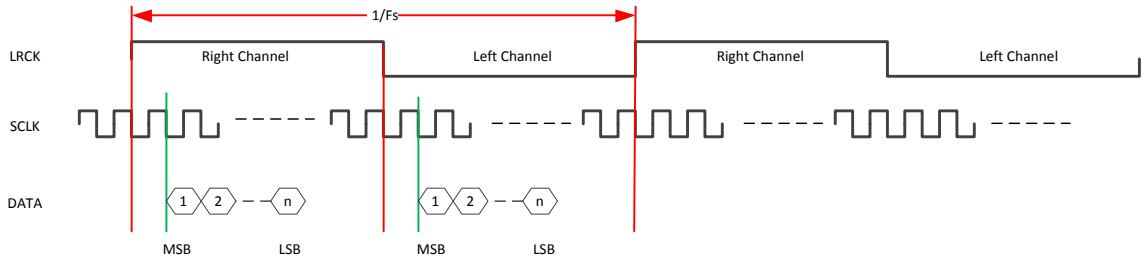


Figure 70: RJ Mode

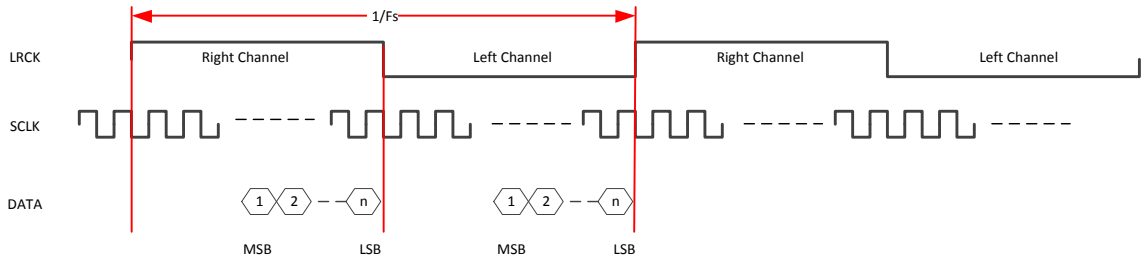


Figure 71: LJ Mode

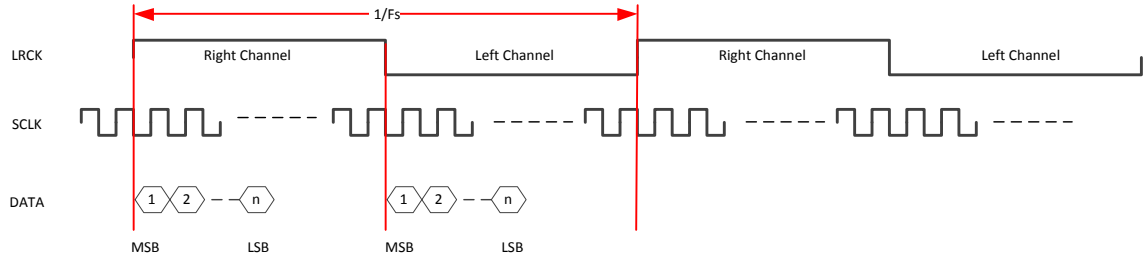
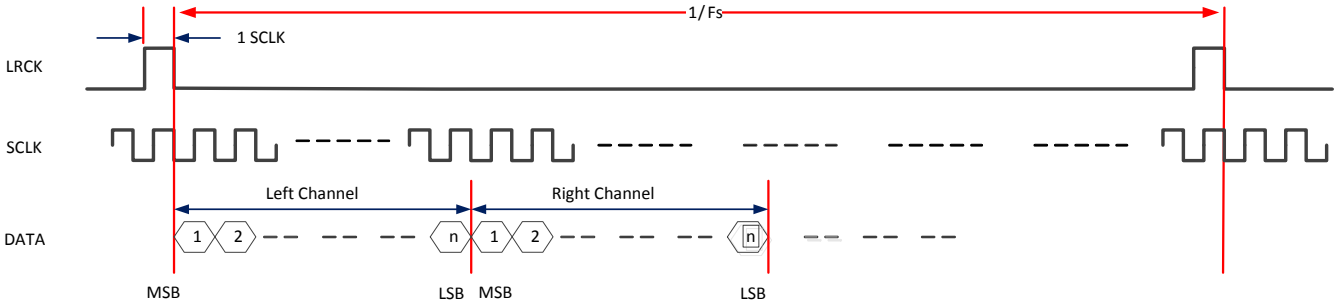
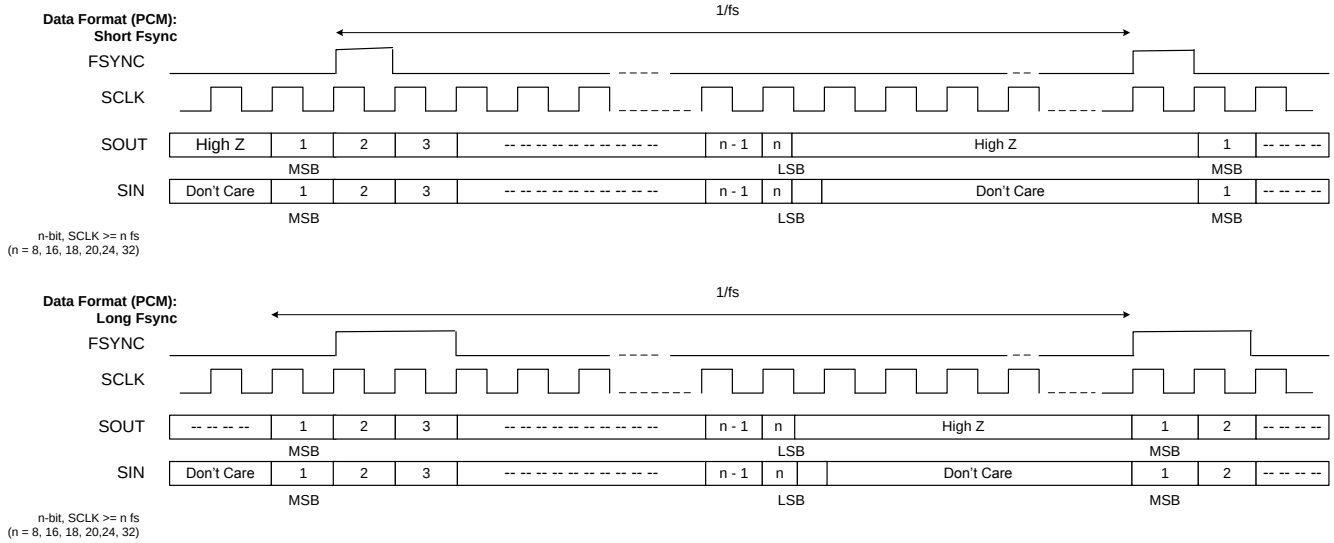


Figure 72: DSP Mode



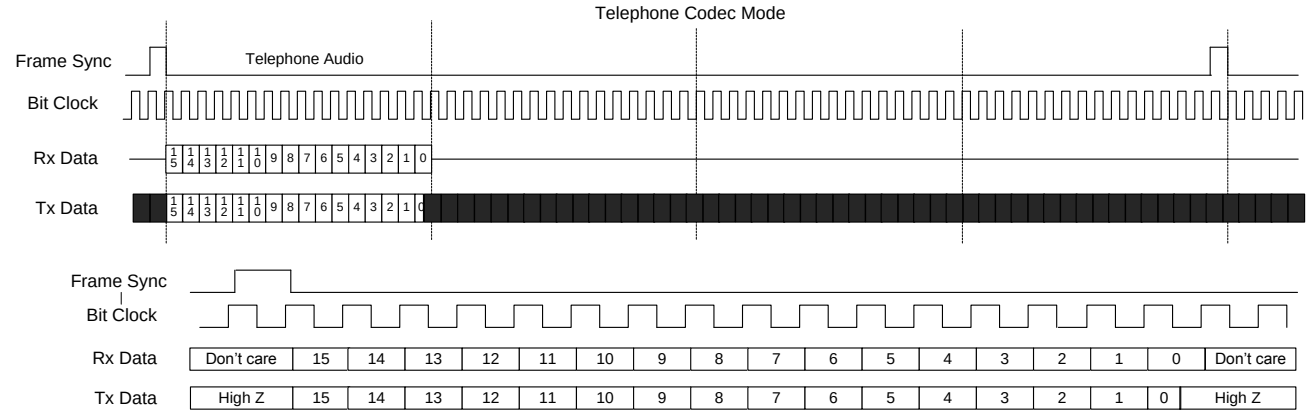
24.1.1.2 Data Formats in PCM Mode

Figure 73: PCM Mode



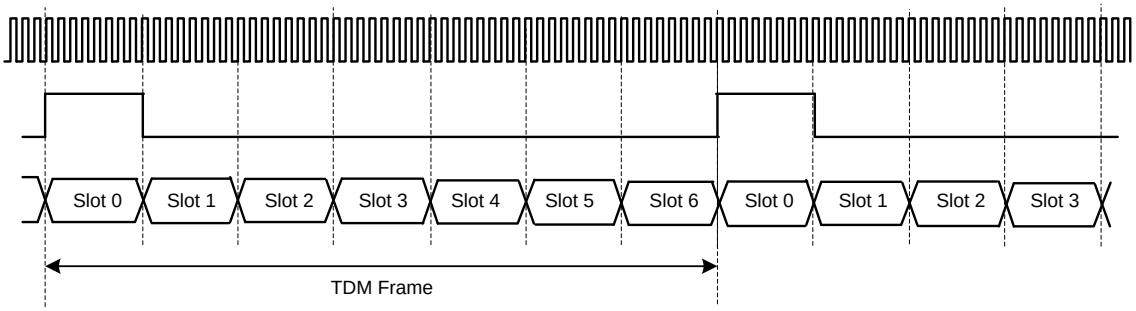
24.1.1.3 Data Formats in Network Mode

Figure 74: Network Mode

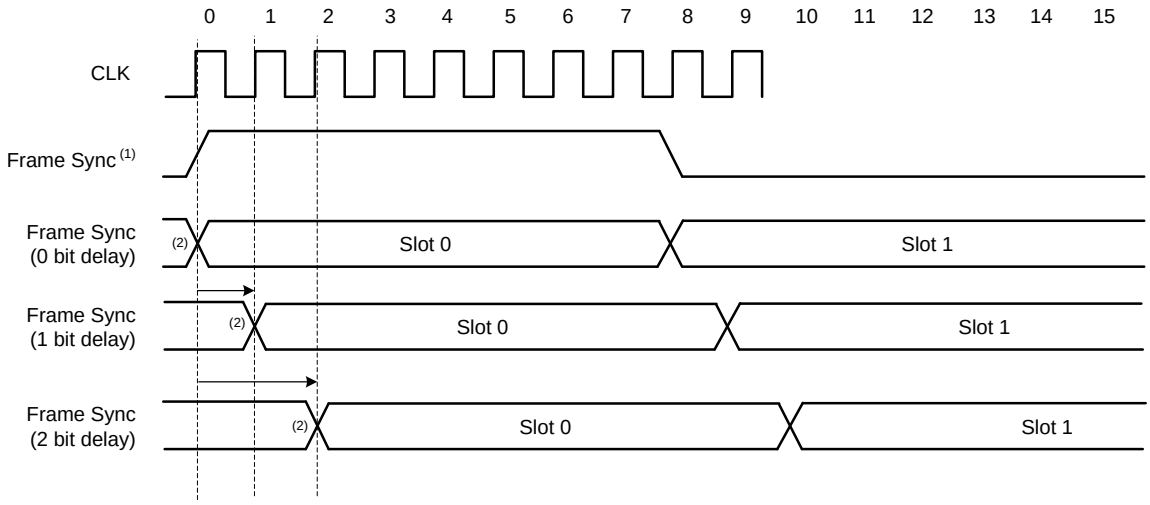


24.1.1.4 Data Format in TDM Mode

Figure 75: TDM Mode



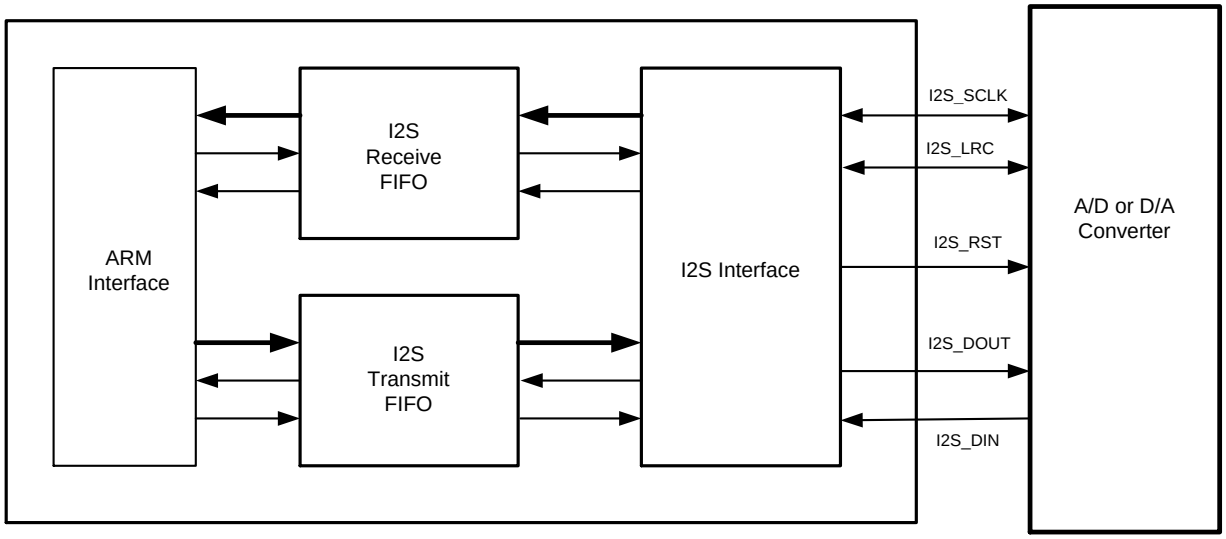
1. FS duration of a slot is shown. FS duration of a single bit is also supported.



1. FS duration of a slot is shown. FS duration of a single bit is also supported.
2. Last bit of the last slot of the previous frame. No gap is allowed between this bit and the first bit of slot 0.

24.1.2 I2S Functionality

Figure 76: I2S Functional Block Diagram



24.1.3 I2S Programming Guidelines

24.1.3.1 Module Initialization

Reset the controller (not mandatory after a POR) and enable the clocks to the controller.

24.1.3.2 I2S Sampling Rate Selection

For LRCK mode, the channel_bit_cnt can be calculated using this equation:

```
Channel_bit_cnt = (frequency of bit_clk) / (2 * required sampling rate) - 1
```

For FSYNC mode, the channel_bit_cnt can be calculated using this equation:

```
Channel_bit_cnt = (frequency of bit_clk) / (required sampling rate) - 1
```

If the above calculation returns a fractional value, the non-symmetry feature of the controller should be used to attain the required sampling rate. In that case, the channel_bit_cnt value should be programmed with an integer that is closest to the fraction, but less than the fraction.

The table below shows some examples for common sampling rates, with CLK_SOURCE_I2S = 24 MHz

Table 65: Common Sampling Rates (CLK_Source_I2S = 24 MHz)

Sampling Rate	I2S_NEW_TIMING[12:00] (mark-space ratio:: Left_channel : Right_channel)				
	CLK_DIVISOR=0 BIT_CLK=24 MHz	CLK_DIVISOR=1 BIT_CLK=12 MHz	CLK_DIVISOR=3 BIT_CLK=6 MHz	CLK_DIVISOR=5 BIT_CLK=4 MHz	CLK_DIVISOR=7 BIT_CLK=3 MHz
8 KHz	0x05DB	0x02ED	0x0176	0x00F9	0x10BB
	(1500:1500)	(750:750)	(375:375)	(250:250)	(187:188)
32 KHz	0x0176	0x10BB	Not supported	0x103E	Not supported
	(375:375)	(187:188)		(62:63)	
44.1 KHz	0x010F	0x0087	0x0043	0x002C	0x0021
	(272:272)	(136:136)	(68:68)	(45:45)	(34:34)
48 KHz	0x00F9	0x007C	0x103E	Not supported	Not supported
	(250:250)	(125:125)	(62:63)		
96 KHz	0x007C	0x103E	0x101F	Not supported	Not supported
	(125:125)	(62:63)	(31:32)		

Note: Ideally, any sampling rate can be generated, either accurately or approximately with any clk_src selected for I2S, if the clk_divisor and the channel_bit_cnt are programmed accordingly.

NON_SYM.EN feature is meant to create the sampling rates approximately, by realizing an odd bit rate with a non-50:50 mark/space ratio. However, if the bit rate (2*channel_bit_cnt) itself is a fraction, the resultant sampling rate will not be accurate. The entries shown as not supported in the above table are attributed to such a deviation.

When the channel_bit_cnt is less than 32, the bit_size should be programmed to be less than channel_bit_cnt.

24.1.3.3 Transmit/Receive Data Flow for I2S Basic, LJM, and RJM Modes

After the module initialization (reset and clock programming), program the corresponding bits of the following registers, preferably in the same sequence:

- **I2S_CTRL:** Set I2S_CTRL[31] = 1/0 to enable/disable I2S Tx mode and I2S_CTRL[30] = 1/0 to enable/disable I2S Rx mode.
- **I2S_CTRL:** Set I2S_CTRL[14:12] = 0 for LRCK mode
- **I2S_OFFSET_0:** Set I2S_OFFSET_0[26:16] = RX_DATA_OFFSET and I2S_OFFSET_0[10:0]= TX_DATA_OFFSET. Settings for this register will decide whether I2S basic, LJM, or RJM mode is selected.
- **I2S_TIMING:** Set I2S_TIMING [10:0] = CHANNEL_BIT_CNT. This is the channel bit count referred to in the “I2S Sampling Rate Selection**”** section.

24.1.3.4 Network, DSP, TDM, and PCM Modes

When in Network, DSP, TDM, or PCM mode, to get the correct fsync, program the following control register bits:

- **I2S_TIMING**: Set I2S_CTRL [10:0] = CHANNEL_BIT_CNT. Refer to the “I2S Sampling Rate Selection**”** section

- **I2S_CTRL:** Set I2S_CTRL[31]=1/0 to enable/disable I2S Tx mode and I2S_CTRL[30]=1/0 to enable/disable I2S Rx mode.
- **I2S_CTRL**: Set I2S_CTRL[14:12]=1 for FSYNC mode
- **I2S_CTRL**: Set I2S_CTRL [9] = 1 to set LRCK_POLARITY as HIGH.
- **I2S_SLOT_CTRL_0**: Set I2S_SLOT_CTRL_0[3:0] = number of slots in a frame - 1.
- **I2S_SLOT_CTRL2_0**: Set I2S_SLOT_CTRL2_0[31:16] for Rx slot enable. Set I2S_SLOT_CTRL2_0[15:0] for Tx slot enable. Each bit represents a slot.

  For example, if bits 17 and 16 are set, it means slot 0 and 1 have been enabled for the Rx path. Similarly, if bit 0 is enabled, it means only slot 0 is enabled for the Tx path.

24.2 Crossbar

Tegra 4 processors have an MxN full crossbar switch with unidirectional data flow, where M and N are the number of TX and RX clients, respectively. A TX client can send audio data to multiple RX clients, while an RX client can receive data from only one TX at a time. A session is created by binding a transmitting TX client and a receiving RX client. At any point, a TX or RX client can be bound to no more than one session. Multiple sessions can be active at a time and work independent of each other.

24.2.1 Clocking

When multiple sessions are in progress, the AHUB clock should be fast enough to support the session with the highest data rate. If a session is used for sending an 8 kHz 16-bit stereo stream, the AHUB clock should be at least 256 kHz. While a faster clock is better for data transmission, it is not necessarily good for power consumption. Using the clock directly from the crystal without going through PLLs is recommended for typical audio applications like MP3 music playback. Depending on the applications, you can program the clock registers to satisfy the requirement of clock frequency. The following table shows the various clock frequencies for AHUB.

Table 66: AHUB Clock Frequencies

Clock Frequency	Description
64 kHz	Slowest clock to carry 8-bit mono voice data in 8 kHz sampling
1.4112 MHz	Clock to carry 16-bit stereo audio data in 44.1 kHz sampling
1.536 MHz	Clock to carry 16-bit stereo audio data in 48 kHz sampling
4.608 MHz	Clock to carry 24-bit stereo audio data in 96 kHz sampling or 16 bit 6 channel audio data in 48 kHz sampling
12 MHz	Minimum crystal frequency
18.432 MHz	Clock to carry 24-bit 8 channel audio data in 96 kHz sampling
36.864 MHz	Clock to carry 24-bit 8 channel audio data in 192 kHz sampling (HDMI maximum)

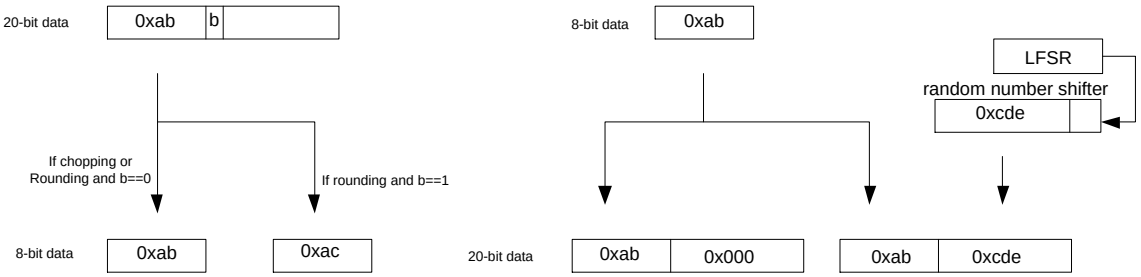
24.2.2 AHUBCIF

The main function of AHUBCIF (generic term for TxCIFs, RxCIFs) is to serialize an audio sample to a serial data stream or deserialize a serial data stream to be an audio sample. The configuration register is programmed to specify the bit width and the number of channels for parsing incoming stream or serializing outgoing stream.

Besides serialization/deserialization, other common format conversions performed in AHUBCIF optionally by the register configuration include data width conversion and channel conversion.

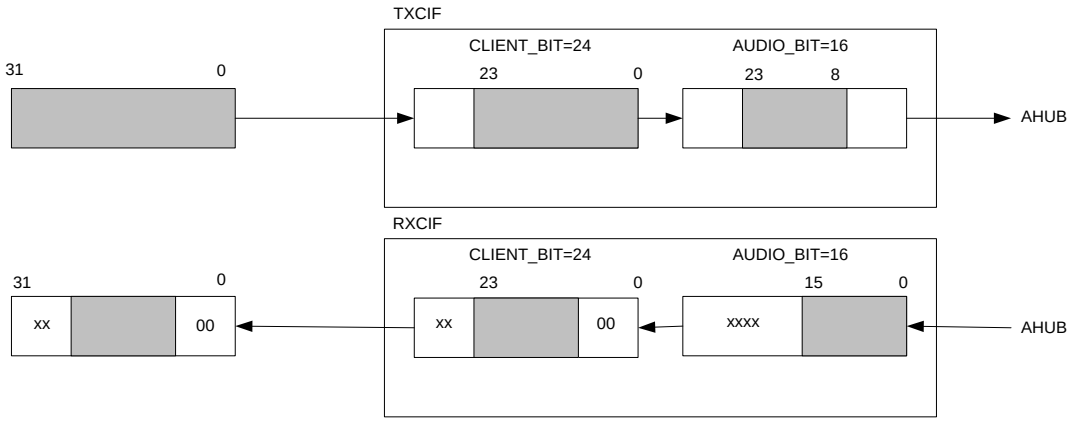
AHUB and AHUBCIF support 8-, 16-, 24-, and 32-bit wide data. When converting from a larger width to a smaller width, a configuration option is provided to chop off the least significant bits and a randomly generated bit can be added to the least significant bit to make audio stream sound more natural. When converting a smaller width to a larger width, the blank region can be either filled with 0’s or 1’s or randomly generated numbers.

Figure 77: Data Format Conversion



AHUBCIF supports mono-to-stereo and stereo-to-mono conversion. Mono data is converted to stereo by duplicating the data into the second channel. The stereo to mono conversion has the following options: copying the first channel data into the mono channel, copying the second channel data into the mono channel, or averaging two channel data, (L + R)/2.

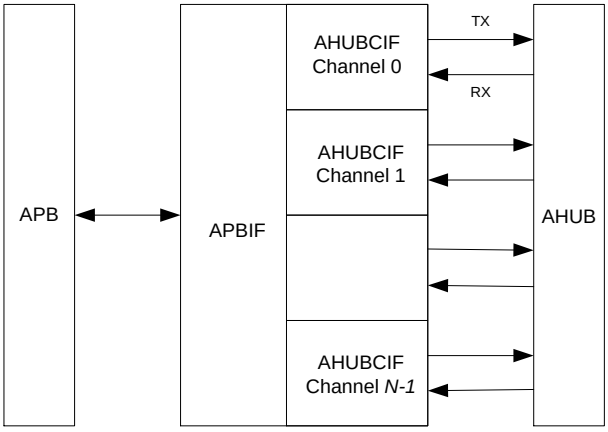
The bit width of audio sample on the Audio HUB side is specified by AUDIO_BITS field. The bit width of audio samples in the client domain is specified by the CLIENT_BITS field. Since AHUBCIF has a 32-bit data interface, the maximum value of CLIENT_BITS and AUDIO_BITS is 32. When the bus is defined to be DATA[31:0], CLIENT_BITS specifies how many bits are valid from LSB in the client. If CLIENT_BITS=24, DATA[23:0] is valid, while DATA[31:24] may be unused. If the client data is serialized with AUDIO_BITS=24, all 24 bits of the client data are transmitted or received through AHUB. However, if AHUB_BITS is less than CLIENT_BITS (for example AHUB_BITS=16), the lower 8 bits are discarded for transmission and only DATA[23:8] will be transmitted to AHUB. The figure below shows the conversion.



24.3 APBIF

APBIF is the agent for the APB control flow and DMA operation, which sends or receives data from/to Memory. APBIF is composed of *N* instances of AHUBIF and APB interface as shown in the figure below.

Figure 78: AHUB Crossbar Switch and Clients



Features

- Separate APB DMA operations for all AHUB clients
- Each channel is bidirectional and needs one TX and one RX AHUBCIF.
- Scalable *N* channel design
- Maintain interrupt registers for AHUB clients

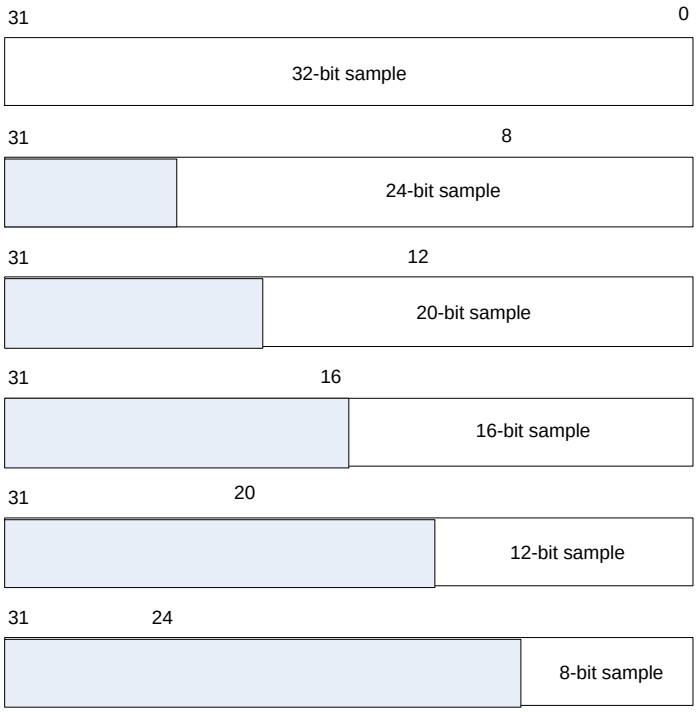
24.3.1 APBIF Functionality

24.3.1.1 Number of Channels

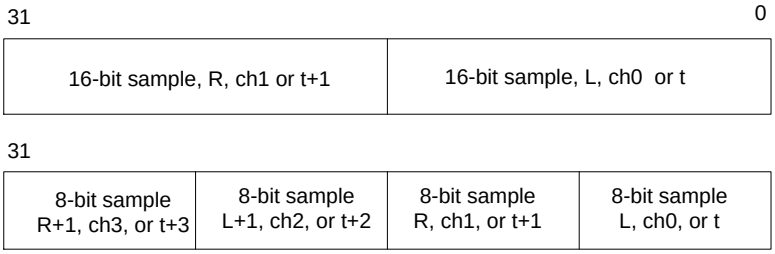
Tegra 4 devices support 10 full-duplex DMA channels. Channels 0-3 reside in the APBIF register space, and Channels 4-9 reside in the APBIF2 register space.

24.3.1.2 Data Packing

When an audio sample is saved in the memory, it is right justified in a Dword configuration, regardless of its bit width. The next figure shows how a sample is stored in the memory without packing.



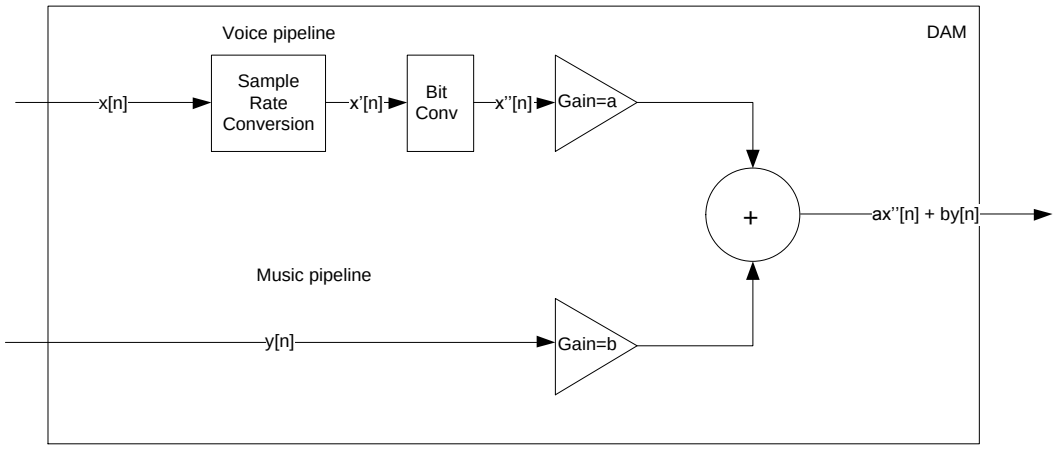
APBIF supports two types of packing, 2x16 bits and 4x8 bits on a Dword boundary. The figure shows the packing modes. When the samples are packed, the sample in the LSB is either the first channel sample (channel 0) or the earlier sample (time T) compared to T+1, T+2 etc. In case of a 4x8 bit configuration, the one in the LSB is the earlier left channel sample or the earliest sample in case of single channel stream.



24.4 Digital Audio Mixer (DAM)

The DAMs have the capability of mixing two input PCM streams of different sampling frequencies and word lengths into one PCM stream. Appropriate gain values can be applied to the individual stream before mixing. The mixing is supported for mono and stereo inputs. A conceptual block diagram of the DAM is shown below.

Figure 79: DAM Block Diagram



In the above figure, the output signals sampling rate and word length will match that of y[n]. The following table shows the possible sample rate conversions.

x[n] \ x'[n]	8 KHz	16 KHz	44.1 KHz	48 KHz
8 KHz	No	Yes	Yes	Yes
16 KHz	Yes	No	Yes	Yes
44.1 KHz	Yes	Yes	No	No
48 KHz	Yes	Yes	No	No

Gain Controller

The gain controller supports a 16-bit volume level and performs a multiply with its input. The gain value is specified in the Q12 unsigned fractional format.

With this format, 0x1000 means 1 in linear gain, while 0x0800 means 0.5 in linear gain. Note that the 24-bit input data is signed data, while the gain value is unsigned.

The accumulator output is shifted right 14 bits to generate a normalized output.

24.4.1 DAM Programming Guidelines

1. Enable APBIF, DAM, and AUDIO clocks and by programing the CLK_RST_CONTROLLER_CLK_OUT_ENB_U register.
2. Enable clocks of other audio modules if DAM is interacting with them.
3. Deassert DAM0 and AUDIO reset by programming CLK_RST_CONTROLLER_RST_DEVICES_U register
4. Deassert reset of other audio modules by similar programming of registers

5. Wait for resets to happen before proceeding further.
6. Start AUDIO programming to connecting CIFs of the Audio Modules involved in the transactions. For example program AUDIO registers AUDIO_I2S0_RX and AUDIO_DAM0_RX to connect the CIFs of I2S and DAM
7. Program Start, high, and low threshold values on I^2S flow control.
8. Start DAM programming
    a. Configure Audio CIF control registers for both the input channels (Rx CIFs) DAM_AUDIOCIF_CH0_CTRL and DAM_AUDIOCIF_CH1_CTRL respectively. Parameters to be configured are MONO_CONV, TRUNCATE, DIRECTION, REPLICATE, STEREO_CONV, EXPAND, CLIENT_BITS, CIF_BITS, CLIENT_CHANNELS, CIF_CHANNELS, and FIFO_THRESHOLD.
    b. Configure the Audio CIF control register DAM_AUDIOCIF_OUT_CTRL for the Tx CIF by the same parameters cited for Rx CIFs.
    c. Set the input FsIn by programming DAM_CH0_CTRL for channel 0 and DAM_CH1_CTRL for channel 1.
    d. Set the Gain for both the channels by setting DAM_CH0_CONV and DAM_CH1_CONV for both channel 0 and channel 1, respectively
    e. Enable DAM output by setting FsOut into the DAM_CTRL register
    f. If the use case involves sample rate conversion either from 44.1 KHz to some frequency or from some frequency to 44.1 KHz, then enable the farrow filter by configuring the DAM_FARROW_PARAM register.
    g. For channel mixing, DATA_SYNC field in both DAM_CH0_CTRL and DAM_CH1_CTRL have to be configured correctly to have mixing in 4 available modes.
        - Ch0 wait on Ch1
        - Ch1 wait on Ch0
        - Both channels wait on each other (Preferred option)
        - Channels don’t wait
    h. Enable DAM by setting the ENABLE bit in DAM_CTRL
9. Program other audio modules by configuring their programmable registers before starting any transactions.

24.5 S/PDIF

The S/PDIF controller supports both input and output in serial audio digital interface format. The input controller can digitally recover a clock from the received stream. The S/PDIF controller is also used to generate the embedded audio for the HDMI output channel.

The controller conforms to the AES/EBU IEC 60958 standard.

The S/PDIF controller consists of two major modules:

- The **S/PDIF Output module**, responsible for sending data to the "spdifout" port in IEC 60958-3 biphase-mark code format
- The **S/PDIF Input module**, responsible for retrieving data to the "spdifin" port in IEC 60958-3 biphase-mark code format

When used for other purposes, this interface is primarily intended to carry audio data coded other than as linear PCM coded audio samples. Provisions are also made to allow the interface to carry data related to computer software or signals coded using non-linear PCM. The format specification for these applications is not part of this standard.

Features

- Fully implements IEC-958 standard interface
- Provides mono and stereo support for sampling frequencies 32 kHz, 44.1 kHz, and 48 kHz

- Supports the following data formats:
  - 16-bit
  - 20-bit
  - 24-bit
  - Raw
  - 16-bit packed
- Supports “autolock” mode to automatically detect the “spdifin” sample rate and lock onto the data stream.
- Supports “override” mode to provide a manual control to sample the “spdifin” data stream.
- Provides a loopback mode to route the “spdifout” back to “spdifin” for self-testing.
- S/PDIF Output (transmitter)
  - 16-word data FIFO for storage of outgoing audio data
  - 4-word user FIFO for storage of outgoing user data
  - 6-word page buffer for storage of outgoing channel status
- S/PDIF Input (receiver)
  - 16-word data FIFO for storage of incoming audio data
  - 4-word user FIFO for storage of incoming user data
  - 6-word page buffer for storage of incoming channel status
- Maximum device clock of 50 MHz

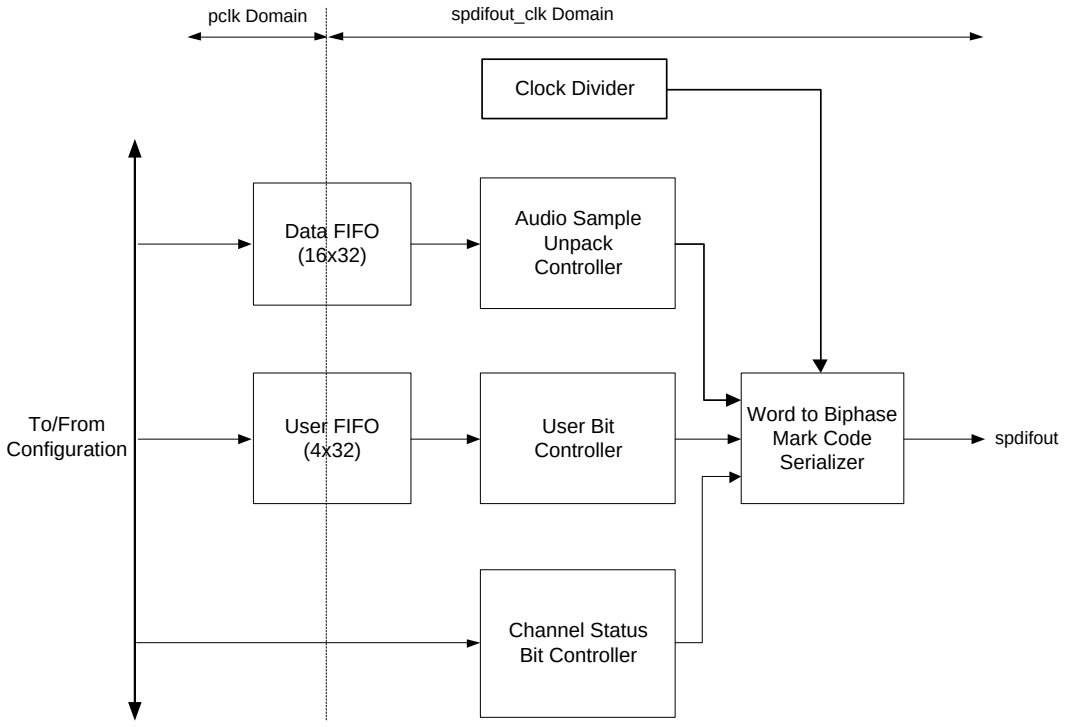
24.5.1 S/PDIF Functionality

24.5.1.1 SPDIFOUT

The S/PDIF Output module contains the following sub-blocks:

- A clock divider
- A 16-word data FIFO
- An audio sample unpack controller
- A 4-word user FIFO
- A user bit controller
- A channel status bit controller
- A word to biphase mark code serializer

Figure 80: SPDIFOUT Block Diagram



Data FIFO

This 16-word deep FIFO is used to store audio samples written by the system in raw mode. It also stores non-audio data such as parity, channel status, user, validity, and preamble bits. This FIFO also serves as a bridge between the 'pclk' and 'spdifout_clk' domains.

Audio Sample Unpack Controller

Based on the PACK register field, this controller either breaks up the packed data from the data FIFO, or lets the Data FIFO's data flow through unmodified to the "word to bi-phase mark code serializer" module. This also controls when to pop an entry out of the data FIFO.

User FIFO

This 4-word deep FIFO is used to store user data written by the system. In raw mode, data from this FIFO is not used. This FIFO also serves as a bridge between the 'pclk' and 'spdifout_clk' domains.

User Bit Controller

When in 16-bit, 20-bit, 24-bit, or pack mode (PACK = 1 or BIT.MODE ≠11) and user data transmit is enabled (TXU.E = 1), this controller temporarily stores an entry out of the user FIFO and feeds user data at the rate of one bit per subframe to the "word to bi-phase mark code serializer" module. This controller also handles when to pop an entry out of the user FIFO. If user data transmit is disabled (TXU.E = 0), the user bit is forced to 0.

Channel Status Bit Controller

When in 16-, 20-, 24-bit, or pack mode (PACK = 1 or BIT.MODE ≠ 11) and channel status transmit is enabled (**TXC.E** = 1), this controller will temporary store an entry out of the channel data TX page buffer and feed channel status, one bit per frame, to

the “word to biphase mark code serializer” module. Also, it provides control to advance the channel data TX page buffer pointer to the next entry. If channel status transmit is disabled (TXC.E = 0), the channel status bit will be forced to 0.

Word to Biphase Mark Code Serializer

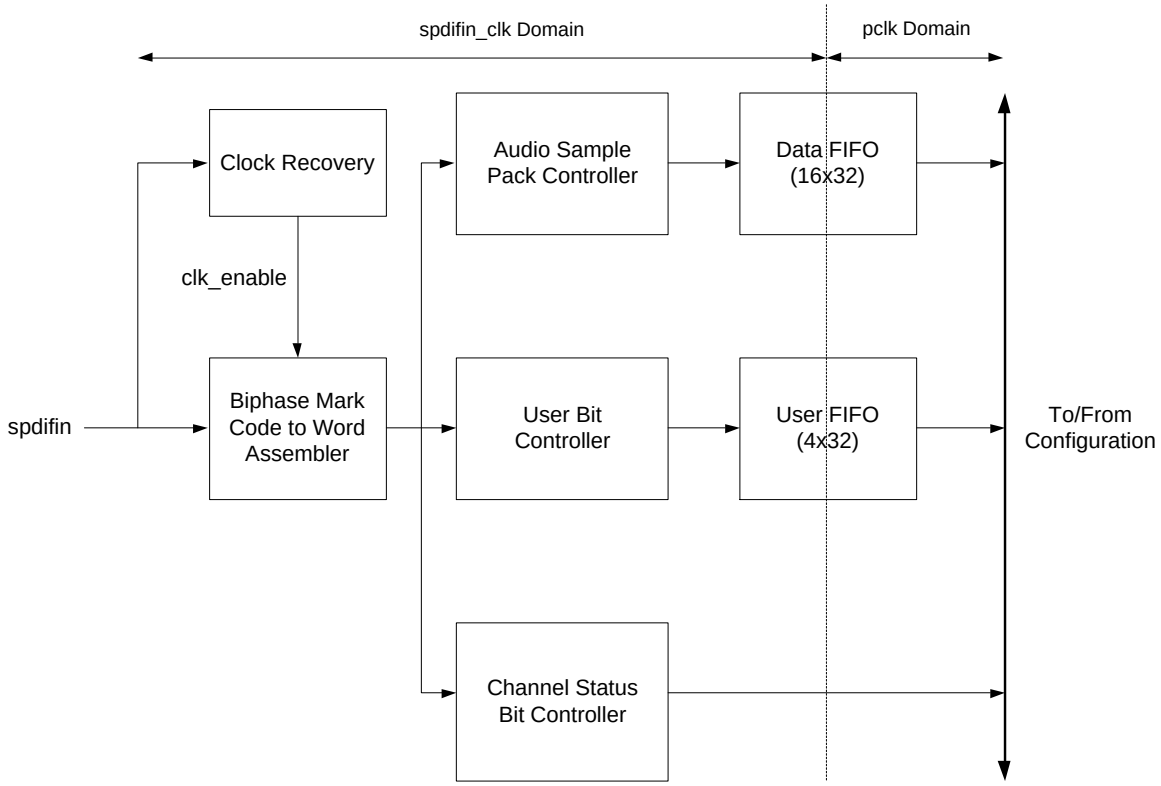
Based on the PACK and BIT.MODE register fields, this controller assembles a 32-bit word from the “Audio sample unpack controller” module, “User bit controller” module, and/or “channel status bit controller” module. This 32-bit word will then be converted to the IEC 60958-3 biphase-mark code format and is sent out through the ‘spdifout’ signal. This controller also provides various status signals (e.g., end of transmitting a subframe) back to the “unpack”, “user”, and “channel status” controllers.

24.5.1.2 SPDIFIN

The S/PDIF Input module contains the following sub-blocks:

- A bi-phase mark code to word assembler
- An audio sample pack controller
- A 16-word data FIFO
- A user bit controller
- A 4-word user FIFO
- A channel status bit controller
- A clock recovery

Figure 81: SPDIFIN Block Diagram



Biphase Mark Code to Word Assembler

Based on the clock enable pulse from the "clock recovery" module, this controller reads the 'spdifin' data stream in IEC 60958-3 biphase mark code format and converts/assembles bit data into a 32-bit word. This 32-bit word is then read by the "audio sample pack controller", "user bit controller", and "channel status bit controller" modules for further processing. This controller also provides various status signals (e.g., end of receiving a subframe) to the "pack", "user", and "channel status" controllers.

Audio Sample Pack Controller

Based on the PACK register field, this controller either combines two audio data or lets the audio data flow through unmodified from the "biphase mark code to word assembler" module to the data FIFO. This also controls when to push an entry into the data FIFO.

Data FIFO

This 16-word deep FIFO is used to store audio samples written by the "audio sample pack controller" module. In 16-bit, 20-bit, 24-bit, or pack mode, it also stores non-audio data such as parity, channel status, user, validity, and preamble bits. This FIFO also serves as a bridge between the 'spdifin_clk' and 'pclk' domains.

User Bit Controller

This controller saves the user bits from each subframe and assembles them into a 32-bit word before pushing an entry into the user FIFO.

User FIFO

This 4-word deep FIFO is used to store user data written by the "user bit controller" module. This FIFO also serves as a bridge between the 'spdifin_clk' and 'pclk' domains.

Channel Status Bit Controller

This controller handles saving the channel status bits from channel A (subframe 1) and assembles them into a 32-bit word before writing into the 6-word channel status page buffer. The channel status page buffer also resides within this controller.

24.5.2 S/PDIF Programming Guidelines

24.5.2.1 16-, 20-, 24-bit Modes

During transmission, the audio data is taken from the TX data FIFO, the channel status bit is taken from the TX channel status page buffer, and the user status bit is from the TX user FIFO. The preamble bits are generated by the hardware. The Valid bit is always zero.

During reception, the audio data is stored in the RX data FIFO, the channel status bit is stored in the RX channel status page buffer and the user status bit is stored in the RX user FIFO.

In addition to storing the audio data, the RX data FIFO also stores the preamble bits, channel status bit, user status bit and the valid bit. The reception of the channel bits start after the B-preamble is detected.

The audio-data sample has 16, 20, or 24 bits of data. Firmware has to transmit 16-, 20-, or 24-bit data and read 16-, 20-, or 24-bit data.

24.5.2.2 Raw Mode

During transmission, the audio data, the preamble bits, channel status bits, the user bits, and the valid bits are transmitted from the RX data FIFO. The firmware has to provide the correct preamble bits for the receiving device to synchronize with the transmitter. The firmware has to provide preambles according to the following table:

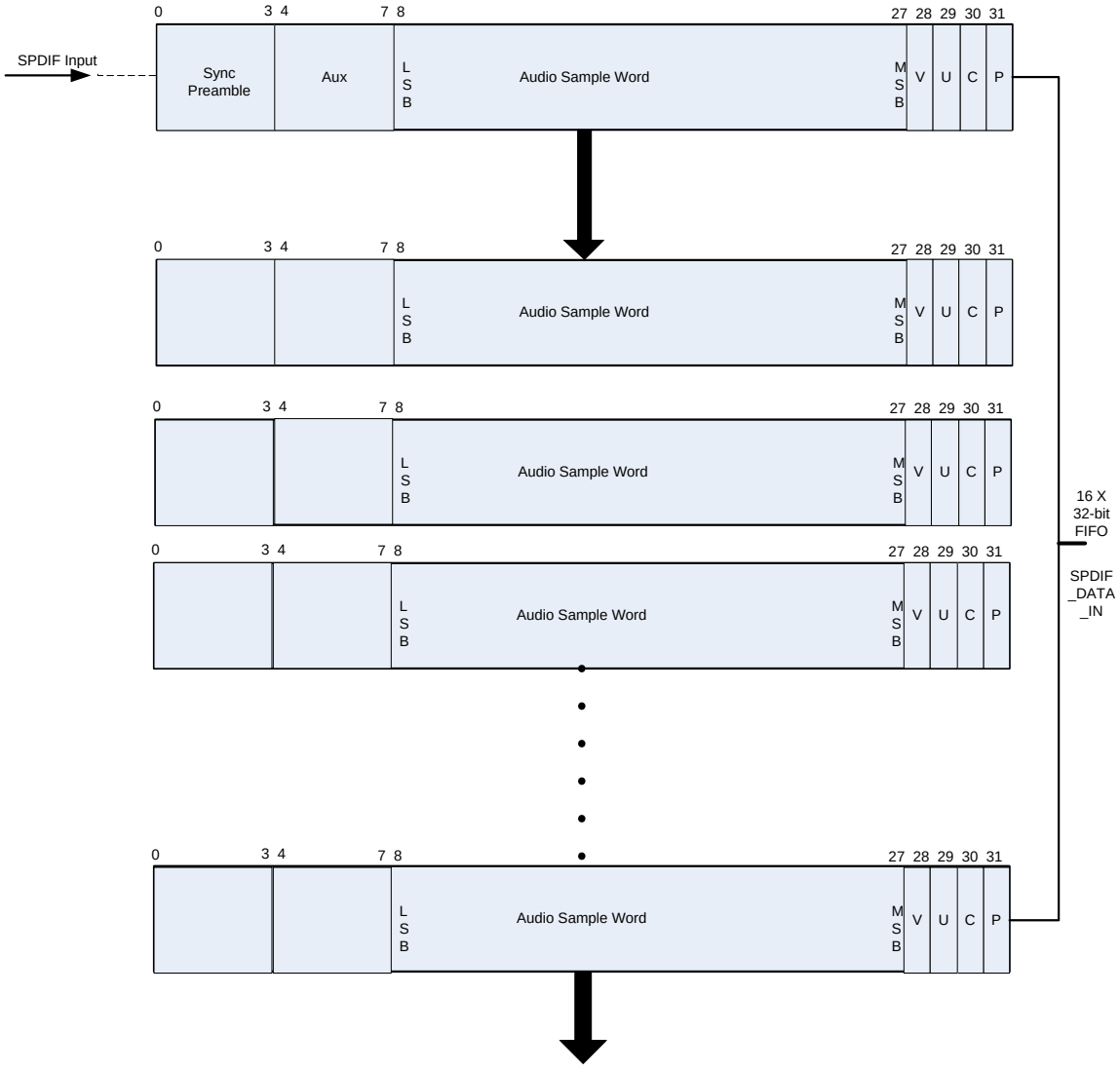
Preceding State in the Parity Bit: 0/1

Table 67: Preamble Bits

Symbol	Preamble Code	Channel Coding	Description
“B” (or “Z”)	0001/0001	11101000/00010111	Start of a block.
“M” (or “X”)	0010/0010	11100010/00011101	Start of sub-frame 1.
“W” (or “Y”)	0011/0011	11100100/00011011	Start of sub-frame 2.

During reception, the audio-data, the preamble bits, channel status bits, the user bits and the valid bits are stored in the RX data FIFO. The user bit is also stored in the RX user FIFO. The channel bit is also stored in the RX channel status page buffer. But the channel status bits are stored in the RX channel status page buffer only after the B-preamble is detected.

Figure 82: S/PDIF Data Format for Raw Mode



24.5.2.3 Pack Mode

During transmission, audio data is taken from TX data FIFO, which contains two 16-bit data (the upper 16 bits are for channel B or sub-frame 2, the lower 16 bits are for channel A or sub-frame 1), the channel status bits are taken from TX channel status page buffer and user status bits taken from TX user FIFO. The preamble bits are generated by the hardware. The valid bit is always zero.

During reception, audio data is stored in the RX data FIFO which contains two 16-bit data (the upper 16 bits are channel B data and the lower 16 bits are channel A data), the channel status bits are stored in the RX channel status page buffer and the user status bits stored in the RX user FIFO. The reception of the channel bits starts after the B-preamble is detected.

24.5.2.4 Module Reset

The application may initialize or reset the S/PDIF module via the clock and reset (CAR) control's RST_DEVICES_L_0 register.

24.5.2.5 Clock Source/Divider Control

The SPDIFOUT clock source/divider can be selected by programming CAR's CLK_SOURCE_SPDIF register. Refer to the S/PDIF registers below for the needed input frequency for the desire output audio sample rate.

The SPDIFIN clock source/divider can be selected by programming CAR's CLK_SOURCE_SPDIF register. Refer to the SPDIF registers below for the needed oversample frequency.

24.5.2.6 Clock Enable Control

The application may enable the S/PDIF clocks by programming CAR's CLK_OUT_ENB_L_0 register.

24.5.2.7 Detecting Incoming Sampling Rate

The SPDIFIN is capable of detecting and locking onto the 'spdifin' data stream regardless of the sample rate. It achieves this by using an oversampling technique.

In addition to reading the sampling rate from the channel status information, one can also read the PERIOD field in the STROBE_CTRL_0 register to estimate the incoming sample rate. The last two digits of PERIOD indicate the fractional clock period.

Sample rate ≈ ('spdifin' clock frequency * 1000) / (PERIOD * 128)

If using the PERIOD method, it is better to read this field toward the end of the song or just before turning off the SPDIFIN. Although the hardware can lock onto the SPDIFIN data stream in as little as 2 subframes of time, the hardware is actually constantly re-adjusting itself (due to jitter, synchronization loss, etc. in the data stream) to provide the strobe point to the center of the biphase data stream. Over time, the strobe point will vary less and less because the hardware has already adjusted to all the variations seen in the data stream.

24.5.2.8 S/PDIF Transmit Data Flow

Make sure that the TX_FIFO is filled before enabling the transmission by setting TX_EN field of CTRL_0 register. Never allow the TX_FIFO to go empty at any time during transmission. The interrupt is generated when the Tx Data FIFO empty count is greater than the Tx DATA attention value (TX_ATN_LVL field of DATA_FIFO_CSR_0 register) if the Tx interrupt enable is set.

24.5.2.9 S/PDIF Receive Data Flow

To enable the reception of RX_FIFO, set the RX_EN field of the CTRL_0 register. Never allow the RX_FIFO to become full at any time during reception. The interrupt is generated when Rx Data FIFO data count is greater than the Rx DATA attention value (RX_ATN_LVL field of the DATA_FIFO_CSR register), if the Rx interrupt enable bit is set.

24.5.2.10 Method of Filling/Retrieve Data from TX/RX User/Data FIFO

Use the DMA method by programming the APB-DMA and APBIF registers. With this method, the FIFO attention levels need to be set in APBIF just like in method 2. But instead of enabling interrupts when the FIFO attention level is reached, one should program the APBDMA registers. Now, every time the attention level is reached, a DMA request will be generated by the APBIF to the APBDMA and the APBDMA will perform the data moving task from/to APBIF FIFOs to the AHB bus.

24.5.3 Clock Rates

The table below shows the clock rates required by each sub-controller for the given sample rate.

Table 68: S/PDIF Clock Rates

Sample Rate (KHz)	S/PDIF Output Clock Frequency (exact frequency MHz)	Period (ns)	S/PDIF Input Clock Frequency (min. frequency but <= 100 MHz)
32.0	4.0960	244.141	16.3840 (48 MHz typical)
44.1	5.6448	177.154	22.5790 (48 MHz typical)
48.0	6.1440	162.760	24.5760 (48 MHz typical)
88.2	11.2896	88.577	45.1584 (72 MHz typical)
96.0	12.2880	81.380	49.1520 (72 MHz typical)
176.4	22.5792	44.289	90.3168 (108 MHz typical)
192.0	24.5760	40.690	98.3040 (108 MHz typical)

24.6 Audio Multiplexer Block (AMX)

The AMX operates in two modes which differ from each other when multiplexing begins:

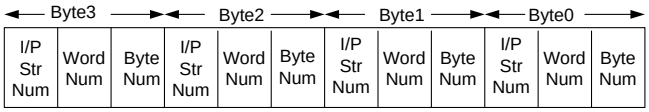
- Mode 0: At the beginning wait for all enabled input streams to have data before forming the very first frame.
- Mode 1: Start whenever data is available in any one of the enabled input streams.

  In either mode, once the first output frame is sent out, the AMX always waits for all active streams to have data available before forming and sending subsequent frames.

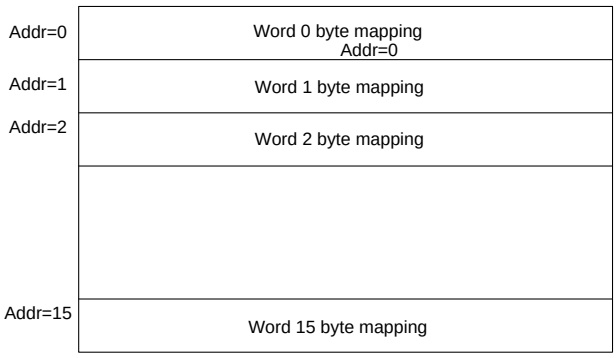
24.6.1 AMX Byte RAM Map

Each byte in the output frame is uniquely mapped to a byte in one of the four input frames. The mapping information is software configurable and is stored in a Byte Map RAM as shown in the figure below.

Figure 83: Byte Map RAM for AMX



Byte mapping for each word in the output frame (for example, Word 0)



Byte Map RAM

The maximum frame size supported is 16 words (or slots) with each word being 32 bits long. The contents of each word in RAM are described in the table below. There are three parameters stored in Byte Map RAM that specify the input source for each output byte.

1. Stream Number: Specifies one of the four input streams that sources the output byte
2. Word number: Specifies the word number within the input stream (as specified in #1 above) frame.
3. Byte number: The byte number within the word specified in #2 above

Table 69: Word Content in RAM

Field	Bits	Description
InputStrNum for output Byte3	31:30	The source stream number of the byte 3.
WordNum for output Byte3	29:26	The word number within source stream specified in bits 31:30
ByteNum for output Byte3	25:24	The byte number within the input frame word specified in 25:24
InputStrNum for output Byte2	23:22	The source stream number of the byte 2.
WordNum for output Byte2	21:18	The word number within source stream specified in bits 21:18
ByteNum for output Byte2	17:16	The byte number within the input frame word specified in 17:16
ChNum for output Byte1	15:14	The source stream number of the byte 1.
WordNum for output Byte1	13:10	The word number within source stream specified in bits 15:14
ByteNum for output Byte1	8:9	The byte number within the input frame word specified in 13:10
InputStrNum for Byte0	7:6	The source stream number of the byte 0.
WordNum for output Byte0	5:2	The word number within source stream specified in bits 5:2
ByteNum for output Byte0	1:0	The byte number within the input frame word specified in 1:0

A few important things to note:

- Input bytes cannot be mapped to output bytes out of order in any stream. This means that word zero in the input frame appears first in the output frame, followed by word 1 and so on.
- Not all bytes within an input frame need be mapped to an output byte.

- Not all output frame bytes need be enabled. Each output byte has a byte enable (software configurable). If a particular byte within output frame is not enabled, the corresponding mapping in the Byte Map RAM is ignored and that byte is populated with zeros

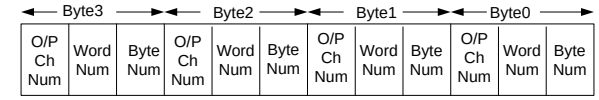
24.6.2 AMX Programming Guidelines

1. Configure Audio CIF control registers for all the four input channels (Rx CIFs) AMX_AUDIOCIF_CH0_CTRL, AMX_AUDIOCIF_CH1_CTRL, AMX_AUDIOCIF_CH2_CTRL and AMX_AUDIOCIF_CH3_CTRL respectively. Parameters to be configured are MONO_CONV, TRUNCATE, DIRECTION, REPLICATE, STEREO_CONV, EXPAND, CLIENT_BITS, AUDIO_BITS, CLIENT_CHANNELS, AUDIO_CHANNELS, and FIFO_THRESHOLD.
2. Configure the **AMX_CTRL_0** register for the **CH_DE**P parameter field which determines whether we need to wait for all enabled channels to have data before transfer or start sending the data when any one of the input channels have data (**WT_ON_ALL & WT_ON_ANY**).
3. Configure the **MSTR_CH_NUM** register for designated master channel, so that if the data is available on particular channel we can send the output.
4. Configure the Audio CIF control register **AMX_AUDIOCIF_OUT_CTRL** for the Tx CIF by the same parameters cited for Rx CIFs.
5. Enable AMX Output by setting **BYTE_EN** in **AMX_OUT_BYTE_EN0** and **AMX_OUT_BYTE_EN1** respectively.
6. Byte RAMCTL programming:
   - For programming RAM with the **HW_ADR_EN** bit unset in the **AMX_AUDIORAMCTL_AMX_CTRL** register, set RAM offset in the **RAM_ADR** field of the **AMX_AUDIORAMCTL_AMX_CTRL** register and sets data in the **DATA** field in **AMX_AUDIORAMCTL_AMX_DATA**.
   - For Programming RAM with the HW_ADR_EN bit set in the **AMX_AUDIORAMCTL_AMX_CTRL** register, set RAM offset in the **RAM_ADR** field of the **AMX_AUDIORAMCTL_AMX_CTRL** register to 0 and set data in the **DATA** field in **AMX_AUDIORAMCTL_AMX_DATA**. Hardware auto-increments **RAM_ADR** after each write.

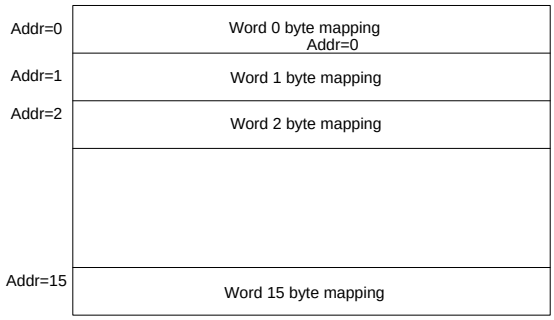
24.7 Demultiplexer Block (ADX)

The demultiplexer block (ADX) takes an input frame and demultiplexes into four audio output channels. Here the functionality is the inverse of the AMX block. Therefore, the Byte Map RAM structure is identical to that of AMX. Each output byte is uniquely mapped to an input channel (unless it is disabled, in which case it will not be used).

Figure 84: Byte Map RAM for ADX



Byte mapping for each word in the output frame (say, Word 0)



Byte Map RAM

Field	Bits	Description
OutputStrNum for output Byte3	31:30	The destination stream number of the byte 3.
WordNum for output Byte3	29:26	The word number within destination stream specified in bits 31:30
ByteNum for output Byte3	25:24	The byte number within the output frame word specified in bits 25:24
OutputStrNum for output Byte2	23:22	The destination stream number of the byte 2.
WordNum for output Byte2	21:18	The word number within destination stream specified in bits 21:18
ByteNum for output Byte2	17:16	The byte number within the output frame word specified in bits 17:16
ChNum for output Byte1	15:14	The destination stream number of the byte 1.
WordNum for output Byte1	13:10	The word number within the destination stream specified in bits 15:14
ByteNum for output Byte1	8:9	The byte number within the output frame word specified in bits 13:10
OutputStrNum for Byte0	7:6	The destination stream number of the byte 0.
WordNum for output Byte0	5:2	The word number within the destination stream specified in bits 5:2
ByteNum for output Byte0	1:0	The byte number within the output frame word specified in bits 1:0

24.7.1 ADX Programming Guidelines

1. Configure Audio CIF control registers for one input channel (Rx CIF) **ADX_AUDIOCIF_IN_CTRL** register. Parameters to be configured are MONO_CONV, TRUNCATE, DIRECTION, REPLICATE, STEREO_CONV, EXPAND, CLIENT_BITS, AUDIO_BITS, CLIENT_CHANNELS, AUDIO_CHANNELS, and FIFO_THRESHOLD.
2. Configure Audio CIF control registers for four output channels **ADX_AUDIOCIF_CH0_CTRL, ADX_AUDIOCIF_CH1_CTRL, ADX_AUDIOCIF_CH2_CTRL** and **ADX_AUDIOCIF_CH3_CTRL** for the Tx CIFs by the same parameters cited for Rx CIF.
3. Configure **ADX_OUT_CH_CTRL_0** for enabling or disabling four output channels.
4. Enable ADX input by setting **BYTE_EN** in **ADX_IN_BYTE_EN0** and **ADX_IN_BYTE_EN1** respectively.
5. Byte RAMCTL:
   - For programming RAM with HW_ADR_EN bit unset in the ADX_AUDIORAMCTL_ADX_CTRL register, set RAM offset in RAM_ADR field of ADX_AUDIORAMCTL_ADX_CTRL register and set data in DATA field in ADX_AUDIORAMCTL_ADX_DATA.
   - For programming RAM with the HW_ADR_EN bit set in the ADX_AUDIORAMCTL_ADX_CTRL register, set RAM offset in RAM_ADR field of ADX_AUDIORAMCTL_ADX_CTRL register to 0 and set data in DATA field in ADX_AUDIORAMCTL_ADX_DATA. Hardware auto-increments the RAM_ADR after each write.

24.8 Audio Hub Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

Control Register for AUDIOCIF for both TX and RX

MONO_CONV : Convert mono to stereo

ZERO -> zero out the second channel
COPY -> copy the first channel to the second channel
valid only in 1 channel to 2 channels conversion

TRUNCATE : rounding method on LSB when deleting lower bits

CHOP -> chop off the lower bits; if 1, just chop off lower bits (truncate)

ROUND -> if immediate lower bit is 1, add 1 to the significant bits

ROUND should be chosen for signed numbers because rounding is toward positive infinite. Also, in case of the max positive number, the lower bits are just chopped off.

For example with 8 -> 4 bit rounding for signed numbers

0x7F -> 0x7

0x0F -> 0x1

0xFF -> 0x0

0x8F -> 0x9

DIRECTION : Direction of CIF

TXCIF -> TX port to AUDIO

RXCIF -> RX port from AUDIO

REPLICATE : When FIFO underflows, if replicate bit is set then previous frame is repeated.

In Tegra 4 devices, only replicating for mono and stereo frames is supported due to the FIFO size limitation. If there are more than 2 channels, the samples in the FIFO are used to fill other multi-channels.

DISABLE -> Disable replication. All zeros will be used.

ENABLE -> Previous frame is repeated.

STEREO_CONV : Convert stereo to mono

CH0 -> pick the first channel

CH1 -> pick the second channel

AVG -> (ch0 + ch1) / 2

valid only in 2 channel to 1 channel conversion

EXPAND : Expand the additional bits with the following method

ZERO -> zero out the expanded field

ONE -> fill the expanded field with 1's

LSFR -> fill the expanded field with a random number

CLIENT_BITS : The number of bits in a sample in the client side

CIF_BITS : The number of bits in a sample in the AUDIO side

CLIENT_CHANNELS : The number of channels in the client side

the actual channels = CLIENT_CHANNELS + 1, i.e., 0 -> 1 channel

CIF_CHANNELS : The number of channels in the AUDIO side

the actual channels = CIF_CHANNELS + 1, i.e., 0 -> 1 channel

FIFO_THRESHOLD : This specifies the number of words that need to be present in the FIFO before a CIF starts transfers.

0 : start transfer when 1 word is received.

1 : start transfer when 2 words are received.

i : start transfer when i words are received.

For example, if FIFO_THRESHOLD is set to 3, then the Rx CIF gives rd_req to the I2S/SPDIF/DAM core only after 4 words have been written into the FIFO. It is recommended that a non-zero value be set only for rx_cifs of I2S and S/PDIF. For all other cases, the default value of zero is the appropriate setting.

24.8.1 APBIF Registers

24.8.1.1 APBIF_CHANNEL0_CTRL_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxx00000000000000000x000x000)

Bit	Reset	Description
31	0x0	TX_ENABLE: 0 = DISABLE 1 = ENABLE
30	0x0	RX_ENABLE: 0 = DISABLE 1 = ENABLE
29	0x0	LOOPBACK: 0 = DISABLE 1 = ENABLE
23:16	0x0	TX_THRESHOLD
15:8	0x0	RX_THRESHOLD
6	0x0	TX_PACK_EN: 0 = DISABLE 1 = ENABLE
5:4	0x0	TX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16
2	0x0	RX_PACK_EN: 0 = DISABLE 1 = ENABLE
1:0	0x0	RX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16

24.8.1.2 APBIF_CHANNEL0_CLEAR_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TX_SOFT_RESET: 0 = DISABLE 1 = ENABLE
30	0x0	RX_SOFT_RESET: 0 = DISABLE 1 = ENABLE

24.8.1.3 APBIF_CHANNEL0_STATUS_0

Offset: 0x8 | Read/Write: RO | Reset: 0xXXXX000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	TX_FREE_COUNT
23:16	X	RX_FREE_COUNT

Bit	Reset	Description
1	X	TX_TRIG: 0 = DISABLE 1 = ENABLE
0	X	RX_TRIG: 0 = DISABLE 1 = ENABLE

24.8.1.4 APBIF_CHANNEL0_TXFIFO_0

Offset: 0xc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

24.8.1.5 APBIF_CHANNEL0_RXFIFO_0

Offset: 0x10 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

24.8.1.6 APBIF_AUDIOCIF_TX0_CTRL_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00))

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14

Bit	R/W	Reset	Description
			14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.1.7 APBIF_AUDIOCIF_RX0_CTRL_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9

Bit	R/W	Reset	Description
			9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP

Bit	R/W	Reset	Description
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.1.8 APBIF_CHANNEL1_CTRL_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxx00000000000000000x000x000)

Bit	Reset	Description
31	0x0	TX_ENABLE: 0 = DISABLE 1 = ENABLE
30	0x0	RX_ENABLE: 0 = DISABLE 1 = ENABLE
29	0x0	LOOPBACK: 0 = DISABLE 1 = ENABLE
23:16	0x0	TX_THRESHOLD
15:8	0x0	RX_THRESHOLD
6	0x0	TX_PACK_EN: 0 = DISABLE 1 = ENABLE
5:4	0x0	TX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16
2	0x0	RX_PACK_EN: 0 = DISABLE 1 = ENABLE
1:0	0x0	RX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16

24.8.1.9 APBIF_CHANNEL1_CLEAR_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TX_SOFT_RESET: 0 = DISABLE 1 = ENABLE
30	0x0	RX_SOFT_RESET: 0 = DISABLE 1 = ENABLE

24.8.1.10 APBIF_CHANNEL1_STATUS_0

Offset: 0x28 | Read/Write: RO | Reset: 0xXXXX000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	TX_FREE_COUNT
23:16	X	RX_FREE_COUNT
1	X	TX_TRIG: 0 = DISABLE 1 = ENABLE
0	X	RX_TRIG: 0 = DISABLE 1 = ENABLE

24.8.1.11 APBIF_CHANNEL1_TXFIFO_0

Offset: 0x2c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

24.8.1.12 APBIF_CHANNEL1_RXFIFO_0

Offset: 0x30 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

24.8.1.13 APBIF_AUDIOCIF_TX1_CTRL_0

Offset: 0x34 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7

Bit	R/W	Reset	Description
			7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.1.14 APBIF_AUDIOCIF_RX1_CTRL_0

Offset: 0x38 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2

Bit	R/W	Reset	Description
			2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.1.15 APBIF_CHANNEL2_CTRL_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxx0000000000000000x000x000)

Bit	Reset	Description
31	0x0	TX_ENABLE: 0 = DISABLE 1 = ENABLE
30	0x0	RX_ENABLE: 0 = DISABLE 1 = ENABLE
29	0x0	LOOPBACK: 0 = DISABLE 1 = ENABLE
23:16	0x0	TX_THRESHOLD
15:8	0x0	RX_THRESHOLD
6	0x0	TX_PACK_EN: 0 = DISABLE 1 = ENABLE
5:4	0x0	TX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16
2	0x0	RX_PACK_EN: 0 = DISABLE 1 = ENABLE
1:0	0x0	RX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16

24.8.1.16 APBIF_CHANNEL2_CLEAR_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TX_SOFT_RESET: 0 = DISABLE 1 = ENABLE
30	0x0	RX_SOFT_RESET: 0 = DISABLE 1 = ENABLE

24.8.1.17 APBIF_CHANNEL2_STATUS_0

Offset: 0x48 | Read/Write: RO | Reset: 0xXXXX000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	TX_FREE_COUNT
23:16	X	RX_FREE_COUNT
1	X	TX_TRIG: 0 = DISABLE 1 = ENABLE
0	X	RX_TRIG: 0 = DISABLE 1 = ENABLE

24.8.1.18 APBIF_CHANNEL2_TXFIFO_0

Offset: 0x4c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

24.8.1.19 APBIF_CHANNEL2_RXFIFO_0

Offset: 0x50 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

24.8.1.20 APBIF_AUDIOCIF_TX2_CTRL_0

Offset: 0x54 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7

Bit	R/W	Reset	Description
			7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.1.21 APBIF_AUDIOCIF_RX2_CTRL_0

Offset: 0x58 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2

Bit	R/W	Reset	Description
			2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.1.22 APBIF_CHANNEL3_CTRL_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxx00000000000000000x000x000)

Bit	Reset	Description
31	0x0	TX_ENABLE: 0 = DISABLE 1 = ENABLE
30	0x0	RX_ENABLE: 0 = DISABLE 1 = ENABLE
29	0x0	LOOPBACK: 0 = DISABLE 1 = ENABLE
23:16	0x0	TX_THRESHOLD
15:8	0x0	RX_THRESHOLD
6	0x0	TX_PACK_EN: 0 = DISABLE 1 = ENABLE
5:4	0x0	TX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16
2	0x0	RX_PACK_EN: 0 = DISABLE 1 = ENABLE
1:0	0x0	RX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16

24.8.1.23 APBIF_CHANNEL3_CLEAR_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TX_SOFT_RESET: 0 = DISABLE 1 = ENABLE
30	0x0	RX_SOFT_RESET: 0 = DISABLE 1 = ENABLE

24.8.1.24 APBIF_CHANNEL3_STATUS_0

Offset: 0x68 | Read/Write: RO | Reset: 0xXXXX000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	TX_FREE_COUNT
23:16	X	RX_FREE_COUNT
1	X	TX_TRIG: 0 = DISABLE 1 = ENABLE
0	X	RX_TRIG: 0 = DISABLE 1 = ENABLE

24.8.1.25 APBIF_CHANNEL3_TXFIFO_0

Offset: 0x6c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

24.8.1.26 APBIF_CHANNEL3_RXFIFO_0

Offset: 0x70 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

24.8.1.27 APBIF_AUDIOCIF_TX3_CTRL_0

Offset: 0x74 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8

Bit	R/W	Reset	Description
			8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP

Bit	R/W	Reset	Description
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.1.28 APBIF_AUDIOCIF_RX3_CTRL_0

Offset: 0x78 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32

Bit	R/W	Reset	Description
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.1.29 APBIF_MISC_CTRL_0

Offset: 0x84 | Read/Write: R/W | Reset: 0xX0000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxx00000)

Bit	R/W	Reset	Description
31	RO	X	AUDIO_ACTIVE: Activity indicator 0 = IDLE 1 = ACTIVE
8	RW	0x0	AUDIO_CG_EN: Clock gating enable 0 = DISABLE 1 = ENABLE

24.8.1.30 APBIF_APBDMA_LIVE_STATUS_0

Offset: 0x88 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CH3_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
30	X	CH3_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
29	X	CH2_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
28	X	CH2_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
27	X	CH1_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26	X	CH1_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
25	X	CH0_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
24	X	CH0_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
23	X	CH3_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
22	X	CH3_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
21	X	CH2_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
20	X	CH2_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
19	X	CH1_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
18	X	CH1_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
17	X	CH0_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
16	X	CH0_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
15	X	CH3_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
14	X	CH3_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
13	X	CH2_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
12	X	CH2_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
11	X	CH1_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
10	X	CH1_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
9	X	CH0_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	X	CH0_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	CH3_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	CH3_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	CH2_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	CH2_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
3	X	CH1_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
2	X	CH1_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	CH0_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	CH0_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

24.8.1.31 APBIF_I2S_LIVE_STATUS_0

Offset: 0x8c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	X	I2S4_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
28	X	I2S4_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
27	X	I2S3_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
26	X	I2S3_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
25	X	I2S2_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
24	X	I2S2_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
23	X	I2S1_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	X	I2S1_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
21	X	I2S0_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
20	X	I2S0_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
19	X	I2S4_RX_ENABLED: 0 = DISABLE 1 = ENABLE
18	X	I2S4_TX_ENABLED: 0 = DISABLE 1 = ENABLE
17	X	I2S3_RX_ENABLED: 0 = DISABLE 1 = ENABLE
16	X	I2S3_TX_ENABLED: 0 = DISABLE 1 = ENABLE
15	X	I2S2_RX_ENABLED: 0 = DISABLE 1 = ENABLE
14	X	I2S2_TX_ENABLED: 0 = DISABLE 1 = ENABLE
13	X	I2S1_RX_ENABLED: 0 = DISABLE 1 = ENABLE
12	X	I2S1_TX_ENABLED: 0 = DISABLE 1 = ENABLE
11	X	I2S0_RX_ENABLED: 0 = DISABLE 1 = ENABLE
10	X	I2S0_TX_ENABLED: 0 = DISABLE 1 = ENABLE
9	X	I2S4_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
8	X	I2S4_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
7	X	I2S3_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	I2S3_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	I2S2_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	X	I2S2_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
3	X	I2S1_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
2	X	I2S1_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	I2S0_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	I2S0_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

24.8.1.32 APBIF_DAM0_LIVE_STATUS_0

Offset: 0x90 | Read/Write: RO | Reset: 0x0X00XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26	X	DAM0_TX_ENABLED: 0 = DISABLE 1 = ENABLE
25	X	DAM0_RX1_ENABLED: 0 = DISABLE 1 = ENABLE
24	X	DAM0_RX0_ENABLED: 0 = DISABLE 1 = ENABLE
15	X	DAM0_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
9	X	DAM0_RX1_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	DAM0_RX0_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	DAM0_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	DAM0_RX1_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	DAM0_RX0_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

24.8.1.33 APBIF_DAM1_LIVE_STATUS_0

Offset: 0x98 | Read/Write: RO | Reset: 0x0X00XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26	X	DAM1_TX_ENABLED: 0 = DISABLE 1 = ENABLE
25	X	DAM1_RX1_ENABLED: 0 = DISABLE 1 = ENABLE
24	X	DAM1_RX0_ENABLED: 0 = DISABLE 1 = ENABLE
15	X	DAM1_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
9	X	DAM1_RX1_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	DAM1_RX0_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	DAM1_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	DAM1_RX1_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	DAM1_RX0_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

24.8.1.34 APBIF_DAM2_LIVE_STATUS_0

Offset: 0xa0 | Read/Write: RO | Reset: 0x0X00XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26	X	DAM2_TX_ENABLED: 0 = DISABLE 1 = ENABLE
25	X	DAM2_RX1_ENABLED: 0 = DISABLE 1 = ENABLE
24	X	DAM2_RX0_ENABLED: 0 = DISABLE 1 = ENABLE
15	X	DAM2_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
9	X	DAM2_RX1_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	DAM2_RX0_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	X	DAM2_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	DAM2_RX1_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	DAM2_RX0_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

24.8.1.35 APBIF_SPDIF_LIVE_STATUS_0

Offset: 0xa8 | Read/Write: RO | Reset: 0x000X0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	SPDIF_TXC_BSY 0 = DISABLE 1 = ENABLE
11	X	SPDIF_USER_TX_ENABLED: 0 = DISABLE 1 = ENABLE
10	X	SPDIF_USER_RX_ENABLED: 0 = DISABLE 1 = ENABLE
9	X	SPDIF_DATA_TX_ENABLED: 0 = DISABLE 1 = ENABLE
8	X	SPDIF_DATA_RX_ENABLED: 0 = DISABLE 1 = ENABLE
7	X	SPDIF_USER_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
6	X	SPDIF_USER_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
5	X	SPDIF_DATA_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
4	X	SPDIF_DATA_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
3	X	SPDIF_USER_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
2	X	SPDIF_USER_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	SPDIF_DATA_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	SPDIF_DATA_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

24.8.1.36 APBIF_I2S_INT_MASK_0

Because the config registers of the AUDIO clients are daisy-chained, their read latency can be high. To avoid unnecessary latency, the clients provide signals to APBI, mostly interrupt signals so that CPU/AVP can access them promptly.

These registers should be detailed further with AUDIO clients.

Interrupt Mask and Status for AUDIO Clients

Offset: 0xb0 | Read/Write: R/W | Reset: 0x7fff03ff (0bx111111111111111xxxxxx1111111111)

Bit	Reset	Description
30	0x1	I2S4_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
29	0x1	I2S3_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
28	0x1	I2S2_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
27	0x1	I2S1_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
26	0x1	I2S0_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
25	0x1	I2S4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
24	0x1	I2S4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
23	0x1	I2S3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
22	0x1	I2S3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
21	0x1	I2S2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
20	0x1	I2S2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
19	0x1	I2S1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
18	0x1	I2S1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
17	0x1	I2S0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
16	0x1	I2S0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
9	0x1	I2S4_RX_DONE: 0 = DISABLE 1 = ENABLE
8	0x1	I2S4_TX_DONE: 0 = DISABLE 1 = ENABLE
7	0x1	I2S3_RX_DONE: 0 = DISABLE 1 = ENABLE
6	0x1	I2S3_TX_DONE: 0 = DISABLE 1 = ENABLE
5	0x1	I2S2_RX_DONE: 0 = DISABLE 1 = ENABLE
4	0x1	I2S2_TX_DONE: 0 = DISABLE 1 = ENABLE
3	0x1	I2S1_RX_DONE: 0 = DISABLE 1 = ENABLE
2	0x1	I2S1_TX_DONE: 0 = DISABLE 1 = ENABLE
1	0x1	I2S0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	0x1	I2S0_TX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.37 APBIF_DAM_INT_MASK_0

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00939393 (0bxxxxxxxxx1xx1xx111xx1xx111xx1xx11)

Bit	Reset	Description
23	0x1	DAM2_CLIP_INTR: 0 = DISABLE 1 = ENABLE
20	0x1	DAM2_TX_DONE: 0 = DISABLE 1 = ENABLE
17	0x1	DAM2_RX1_DONE: 0 = DISABLE 1 = ENABLE
16	0x1	DAM2_RX0_DONE: 0 = DISABLE 1 = ENABLE
15	0x1	DAM1_CLIP_INTR: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	0x1	DAM1_TX_DONE: 0 = DISABLE 1 = ENABLE
9	0x1	DAM1_RX1_DONE: 0 = DISABLE 1 = ENABLE
8	0x1	DAM1_RX0_DONE: 0 = DISABLE 1 = ENABLE
7	0x1	DAM0_CLIP_INTR: 0 = DISABLE 1 = ENABLE
4	0x1	DAM0_TX_DONE: 0 = DISABLE 1 = ENABLE
1	0x1	DAM0_RX1_DONE: 0 = DISABLE 1 = ENABLE
0	0x1	DAM0_RX0_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.38 APBIF_SPDIF_INT_MASK_0

Offset: 0xbc | Read/Write: R/W | Reset: 0x00008fff (0bxxxxxxxxxxxxxxxx1xxx111111111111)

Bit	Reset	Description
15	0x1	SPDIF_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
11	0x1	SPDIF_BAD_PREAMBLE: 0 = DISABLE 1 = ENABLE
10	0x1	SPDIF_B_SYNC: 0 = DISABLE 1 = ENABLE
9	0x1	SPDIF_RX_CHANNEL: 0 = DISABLE 1 = ENABLE
8	0x1	SPDIF_RX_IU: 0 = DISABLE 1 = ENABLE
7	0x1	SPDIF_USER_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x1	SPDIF_USER_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x1	SPDIF_DATA_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x1	SPDIF_DATA_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x1	SPDIF_USER_TX_DONE: 0 = DISABLE 1 = ENABLE
2	0x1	SPDIF_USER_RX_DONE: 0 = DISABLE 1 = ENABLE
1	0x1	SPDIF_DATA_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x1	SPDIF_DATA_RX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.39 APBIF_APBIF_INT_MASK_0

Offset: 0xc0 | Read/Write: R/W | Reset: 0x0001ffff (0bxxxxxxxxxxxxxxxx11111111111111111)

Bit	Reset	Description
16	0x1	CONFIG_LINK_TIMEOUT: 0 = DISABLE 1 = ENABLE
15	0x1	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
14	0x1	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
13	0x1	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
12	0x1	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
11	0x1	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
10	0x1	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
9	0x1	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x1	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x1	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x1	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x1	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x1	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x1	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
2	0x1	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x1	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x1	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.40 APBIF_I2S_INT_STATUS_0

Bits in this register are high after the interrupt condition is satisfied. A write to this register clears the bits corresponding to the data bits that are high in the write data.

Interrupt Status Registers

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000xxxxxx0000000000)

Bit	Reset	Description
30	0x0	I2S4_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
29	0x0	I2S3_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
28	0x0	I2S2_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
27	0x0	I2S1_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
26	0x0	I2S0_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
25	0x0	I2S4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
24	0x0	I2S4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
23	0x0	I2S3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
22	0x0	I2S3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
21	0x0	I2S2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
20	0x0	I2S2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
19	0x0	I2S1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
18	0x0	I2S1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
17	0x0	I2S0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
16	0x0	I2S0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
9	0x0	I2S4_RX_DONE: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX_DONE: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_RX_DONE: 0 = DISABLE 1 = ENABLE
6	0x0	I2S3_TX_DONE: 0 = DISABLE 1 = ENABLE
5	0x0	I2S2_RX_DONE: 0 = DISABLE 1 = ENABLE
4	0x0	I2S2_TX_DONE: 0 = DISABLE 1 = ENABLE
3	0x0	I2S1_RX_DONE: 0 = DISABLE 1 = ENABLE
2	0x0	I2S1_TX_DONE: 0 = DISABLE 1 = ENABLE
1	0x0	I2S0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	I2S0_TX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.41 APBIF_DAM_INT_STATUS_0

Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx0xx0xx000xx0xx000xx0xx00)

Bit	Reset	Description
23	0x0	DAM2_CLIP_INTR: 0 = DISABLE 1 = ENABLE
20	0x0	DAM2_TX_DONE: 0 = DISABLE 1 = ENABLE
17	0x0	DAM2_RX1_DONE: 0 = DISABLE 1 = ENABLE
16	0x0	DAM2_RX0_DONE: 0 = DISABLE 1 = ENABLE
15	0x0	DAM1_CLIP_INTR: 0 = DISABLE 1 = ENABLE
12	0x0	DAM1_TX_DONE: 0 = DISABLE 1 = ENABLE
9	0x0	DAM1_RX1_DONE: 0 = DISABLE 1 = ENABLE
8	0x0	DAM1_RX0_DONE: 0 = DISABLE 1 = ENABLE
7	0x0	DAM0_CLIP_INTR: 0 = DISABLE 1 = ENABLE
4	0x0	DAM0_TX_DONE: 0 = DISABLE 1 = ENABLE
1	0x0	DAM0_RX1_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	DAM0_RX0_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.42 APBIF_SPDIF_INT_STATUS_0

Offset: 0xd4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0xxx000000000000)

Bit	Reset	Description
15	0x0	SPDIF_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
11	0x0	SPDIF_BAD_PREAMBLE: 0 = DISABLE 1 = ENABLE
10	0x0	SPDIF_B_SYNC: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	0x0	SPDIF_RX_CHANNEL: 0 = DISABLE 1 = ENABLE
8	0x0	SPDIF_RX_IU: 0 = DISABLE 1 = ENABLE
7	0x0	SPDIF_USER_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	SPDIF_USER_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	SPDIF_DATA_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	SPDIF_DATA_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	SPDIF_USER_TX_DONE: 0 = DISABLE 1 = ENABLE
2	0x0	SPDIF_USER_RX_DONE: 0 = DISABLE 1 = ENABLE
1	0x0	SPDIF_DATA_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	SPDIF_DATA_RX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.43 APBIF_APBIF_INT_STATUS_0

Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
16	0x0	CONFIG_LINK_TIMEOUT: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
13	0x0	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
12	0x0	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
11	0x0	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10	0x0	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
9	0x0	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.44 APBIF_I2S_INT_SOURCE_0

Interrupt Source

Offset: 0xe0 | Read/Write: RO | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	I2S4_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
29	X	I2S3_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
28	X	I2S2_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
27	X	I2S1_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26	X	I2S0_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
25	X	I2S4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
24	X	I2S4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
23	X	I2S3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
22	X	I2S3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
21	X	I2S2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
20	X	I2S2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
19	X	I2S1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
18	X	I2S1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
17	X	I2S0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
16	X	I2S0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
9	X	I2S4_RX_DONE: 0 = DISABLE 1 = ENABLE
8	X	I2S4_TX_DONE: 0 = DISABLE 1 = ENABLE
7	X	I2S3_RX_DONE: 0 = DISABLE 1 = ENABLE
6	X	I2S3_TX_DONE: 0 = DISABLE 1 = ENABLE
5	X	I2S2_RX_DONE: 0 = DISABLE 1 = ENABLE
4	X	I2S2_TX_DONE: 0 = DISABLE 1 = ENABLE
3	X	I2S1_RX_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	X	I2S1_TX_DONE: 0 = DISABLE 1 = ENABLE
1	X	I2S0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	X	I2S0_TX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.45 APBIF_DAM_INT_SOURCE_0

Offset: 0xe4 | Read/Write: RO | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23	X	DAM2_CLIP_INTR: 0 = DISABLE 1 = ENABLE
20	X	DAM2_TX_DONE: 0 = DISABLE 1 = ENABLE
17	X	DAM2_RX1_DONE: 0 = DISABLE 1 = ENABLE
16	X	DAM2_RX0_DONE: 0 = DISABLE 1 = ENABLE
15	X	DAM1_CLIP_INTR: 0 = DISABLE 1 = ENABLE
12	X	DAM1_TX_DONE: 0 = DISABLE 1 = ENABLE
9	X	DAM1_RX1_DONE: 0 = DISABLE 1 = ENABLE
8	X	DAM1_RX0_DONE: 0 = DISABLE 1 = ENABLE
7	X	DAM0_CLIP_INTR: 0 = DISABLE 1 = ENABLE
4	X	DAM0_TX_DONE: 0 = DISABLE 1 = ENABLE
1	X	DAM0_RX1_DONE: 0 = DISABLE 1 = ENABLE
0	X	DAM0_RX0_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.46 APBIF_SPDIF_INT_SOURCE_0

Offset: 0xec | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15	X	SPDIF_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
11	X	SPDIF_BAD_PREAMBLE: 0 = DISABLE 1 = ENABLE
10	X	SPDIF_B_SYNC: 0 = DISABLE 1 = ENABLE
9	X	SPDIF_RX_CHANNEL: 0 = DISABLE 1 = ENABLE
8	X	SPDIF_RX_IU: 0 = DISABLE 1 = ENABLE
7	X	SPDIF_USER_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	X	SPDIF_USER_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	X	SPDIF_DATA_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	X	SPDIF_DATA_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	X	SPDIF_USER_TX_DONE: 0 = DISABLE 1 = ENABLE
2	X	SPDIF_USER_RX_DONE: 0 = DISABLE 1 = ENABLE
1	X	SPDIF_DATA_TX_DONE: 0 = DISABLE 1 = ENABLE
0	X	SPDIF_DATA_RX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.47 APBIF_APBIF_INT_SOURCE_0

Offset: 0xf0 | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
16	X	CONFIG_LINK_TIMEOUT: 0 = DISABLE 1 = ENABLE
15	X	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
14	X	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
13	X	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
12	X	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
11	X	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
10	X	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
9	X	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
8	X	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	X	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	X	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	X	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	X	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	X	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
2	X	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	X	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	X	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.48 APBIF_I2S_INT_SET_0

Interrupt Set

Offset: 0xf8 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxx0000000000)

Bit	Reset	Description
30	0x0	I2S4_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
29	0x0	I2S3_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
28	0x0	I2S2_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
27	0x0	I2S1_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
26	0x0	I2S0_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
25	0x0	I2S4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
24	0x0	I2S4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
23	0x0	I2S3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
22	0x0	I2S3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
21	0x0	I2S2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
20	0x0	I2S2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
19	0x0	I2S1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
18	0x0	I2S1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
17	0x0	I2S0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
16	0x0	I2S0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
9	0x0	I2S4_RX_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	I2S4_TX_DONE: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_RX_DONE: 0 = DISABLE 1 = ENABLE
6	0x0	I2S3_TX_DONE: 0 = DISABLE 1 = ENABLE
5	0x0	I2S2_RX_DONE: 0 = DISABLE 1 = ENABLE
4	0x0	I2S2_TX_DONE: 0 = DISABLE 1 = ENABLE
3	0x0	I2S1_RX_DONE: 0 = DISABLE 1 = ENABLE
2	0x0	I2S1_TX_DONE: 0 = DISABLE 1 = ENABLE
1	0x0	I2S0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	I2S0_TX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.49 APBIF_DAM_INT_SET_0

Offset: 0xfc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx0xx0xx000xx0xx000xx0xx00)

Bit	Reset	Description
23	0x0	DAM2_CLIP_INTR: 0 = DISABLE 1 = ENABLE
20	0x0	DAM2_TX_DONE: 0 = DISABLE 1 = ENABLE
17	0x0	DAM2_RX1_DONE: 0 = DISABLE 1 = ENABLE
16	0x0	DAM2_RX0_DONE: 0 = DISABLE 1 = ENABLE
15	0x0	DAM1_CLIP_INTR: 0 = DISABLE 1 = ENABLE
12	0x0	DAM1_TX_DONE: 0 = DISABLE 1 = ENABLE
9	0x0	DAM1_RX1_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	DAM1_RX0_DONE: 0 = DISABLE 1 = ENABLE
7	0x0	DAM0_CLIP_INTR: 0 = DISABLE 1 = ENABLE
4	0x0	DAM0_TX_DONE: 0 = DISABLE 1 = ENABLE
1	0x0	DAM0_RX1_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	DAM0_RX0_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.50 APBIF_SPDIF_INT_SET_0

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxx000000000000)

Bit	Reset	Description
15	0x0	SPDIF_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
11	0x0	SPDIF_BAD_PREAMBLE: 0 = DISABLE 1 = ENABLE
10	0x0	SPDIF_B_SYNC: 0 = DISABLE 1 = ENABLE
9	0x0	SPDIF_RX_CHANNEL: 0 = DISABLE 1 = ENABLE
8	0x0	SPDIF_RX_IU: 0 = DISABLE 1 = ENABLE
7	0x0	SPDIF_USER_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	SPDIF_USER_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	SPDIF_DATA_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	SPDIF_DATA_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	SPDIF_USER_TX_DONE: 0 = DISABLE 1 = ENABLE
2	0x0	SPDIF_USER_RX_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	SPDIF_DATA_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	SPDIF_DATA_RX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.51 APBIF_APBIF_INT_SET_0

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx00000000000000000)

Bit	Reset	Description
16	0x0	CONFIG_LINK_TIMEOUT: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
13	0x0	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
12	0x0	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
11	0x0	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
10	0x0	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
9	0x0	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.52 APBIF_APBDMA_LIVE_TX_DMA_FIFO_EMPTY_STATUS_0

Offset: 0x10c │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
8	X	CH8_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
7	X	CH7_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	CH6_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	CH5_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	CH4_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
3	X	CH3_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
2	X	CH2_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	CH1_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	CH0_TX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

24.8.1.53 APBIF_APBDMA_LIVE_TX_CIF_FIFO_EMPTY_STATUS_0

Offset: 0x110 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	X	CH8_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
7	X	CH7_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	CH6_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	CH5_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	CH4_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
3	X	CH3_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
2	X	CH2_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	CH1_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	CH0_TX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

24.8.1.54 APBIF_APBDMA_LIVE_RX_DMA_FIFO_EMPTY_STATUS_0

Offset: 0x114 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
8	X	CH8_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
7	X	CH7_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	CH6_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	CH5_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	CH4_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
3	X	CH3_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	X	CH2_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	CH1_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	CH0_RX_DMA_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

24.8.1.55 APBIF_APBDMA_LIVE_RX_CIF_FIFO_EMPTY_STATUS_0

Offset: 0x118 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
8	X	CH8_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
7	X	CH7_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	CH6_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	CH5_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	CH4_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
3	X	CH3_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
2	X	CH2_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
1	X	CH1_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	CH0_RX_CIF_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

24.8.1.56 APBIF_APBDMA_LIVE_TX_DMA_FIFO_FULL_STATUS_0

Offset: 0x11c │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	X	CH8_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	CH7_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
6	X	CH6_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
5	X	CH5_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
4	X	CH4_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
3	X	CH3_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
2	X	CH2_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
1	X	CH1_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
0	X	CH0_TX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE

24.8.1.57 APBIF_APBDMA_LIVE_TX_CIF_FIFO_FULL_STATUS_0

Offset: 0x120 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	CH8_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	CH7_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
6	X	CH6_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
5	X	CH5_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
4	X	CH4_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
3	X	CH3_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	X	CH2_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
1	X	CH1_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
0	X	CH0_TX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE

24.8.1.58 APBIF_APBDMA_LIVE_RX_DMA_FIFO_FULL_STATUS_0

Offset: 0x124 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	CH8_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	CH7_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
6	X	CH6_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
5	X	CH5_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
4	X	CH4_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
3	X	CH3_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
2	X	CH2_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
1	X	CH1_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE
0	X	CH0_RX_DMA_FIFO_FULL: 0 = DISABLE 1 = ENABLE

24.8.1.59 APBIF_APBDMA_LIVE_RX_CIF_FIFO_FULL_STATUS_0

Offset: 0x128 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	CH9_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	X	CH8_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	CH7_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
6	X	CH6_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
5	X	CH5_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
4	X	CH4_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
3	X	CH3_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
2	X	CH2_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
1	X	CH1_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE
0	X	CH0_RX_CIF_FIFO_FULL: 0 = DISABLE 1 = ENABLE

24.8.1.60 APBIF_APBIF_RXCIF_UNDERRUN_INT_MASK_0

Offset: 0x12c │ Read/Write: R/W │ Reset: 0x000003ff (0bxxxxxxxxxxxxxxxxxxxxxxxx1111111111)

Bit	Reset	Description
9	0x1	APBIF_CH9_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
8	0x1	APBIF_CH8_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x1	APBIF_CH7_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	0x1	APBIF_CH6_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x1	APBIF_CH5_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	0x1	APBIF_CH4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x1	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x1	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x1	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	0x1	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.61 APBIF_APBIF_TXCIF_OVERRUN_INT_MASK_0

Offset: 0x130 │ Read/Write: R/W │ Reset: 0x000003ff (0bxxxxxxxxxxxxxxxxxxxxxxxx1111111111)

Bit	Reset	Description
9	0x1	APBIF_CH9_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x1	APBIF_CH8_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
7	0x1	APBIF_CH7_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x1	APBIF_CH6_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
5	0x1	APBIF_CH5_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x1	APBIF_CH4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
3	0x1	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
2	0x1	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
1	0x1	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x1	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.62 APBIF_APBIF_APBDMA_UNDERRUN_INT_MASK_0

Offset: 0x134 │ Read/Write: R/W │ Reset: 0x000003ff (0bxxxxxxxxxxxxxxxxxxxxxxxx1111111111)

Bit	Reset	Description
9	0x1	APBIF_CH9_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x1	APBIF_CH8_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x1	APBIF_CH7_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	0x1	APBIF_CH6_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x1	APBIF_CH5_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	0x1	APBIF_CH4_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x1	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
2	0x1	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x1	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	0x1	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.63 APBIF_APBIF_APBDMA_OVERRUN_INT_MASK_0

Offset: 0x138 │ Read/Write: R/W │ Reset: 0x000003ff (0bxxxxxxxxxxxxxxxxxxxxxxxx1111111111)

Bit	Reset	Description
9	0x1	APBIF_CH9_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x1	APBIF_CH8_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
7	0x1	APBIF_CH7_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x1	APBIF_CH6_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
5	0x1	APBIF_CH5_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x1	APBIF_CH4_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
3	0x1	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x1	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
1	0x1	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x1	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.64 APBIF_APBIF_RXCIF_UNDERRUN_INT_STATUS_0

Offset: 0x13c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH8_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.65 APBIF_APBIF_TXCIF_OVERRUN_INT_STATUS_0

Offset: 0x140 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	APBIF_CH8_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.66 APBIF_APBIF_APBDMA_UNDERRUN_INT_STATUS_0

Offset: 0x144 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH8_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.67 APBIF_APBIF_APBDMA_OVERRUN_INT_STATUS_0

Offset: 0x148 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH8_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.68 APBIF_APBIF_RXCIF_UNDERRUN_INT_SOURCE_0

Offset: 0x14c │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	APBIF_CH9_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	X	APBIF_CH8_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	X	APBIF_CH7_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	X	APBIF_CH6_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	X	APBIF_CH5_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	X	APBIF_CH4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	X	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
2	X	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	X	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	X	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.69 APBIF_APBIF_TXCIF_OVERRUN_INT_SOURCE_0

Offset: 0x150 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	APBIF_CH9_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
8	X	APBIF_CH8_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
7	X	APBIF_CH7_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	X	APBIF_CH6_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
5	X	APBIF_CH5_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	X	APBIF_CH4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
3	X	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	X	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
1	X	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	X	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.70 APBIF_APBIF_APBDMA_UNDERRUN_INT_SOURCE_0

Offset: 0x154 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	APBIF_CH9_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
8	X	APBIF_CH8_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	X	APBIF_CH7_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	X	APBIF_CH6_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	X	APBIF_CH5_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	X	APBIF_CH4_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	X	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
2	X	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	X	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	X	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.71 APBIF_APBIF_APBDMA_OVERRUN_INT_SOURCE_0

Offset: 0x158 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	APBIF_CH9_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	X	APBIF_CH8_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
7	X	APBIF_CH7_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
6	X	APBIF_CH6_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
5	X	APBIF_CH5_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
4	X	APBIF_CH4_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
3	X	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
2	X	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
1	X	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
0	X	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.72 APBIF_APBIF_RXCIF_UNDERRUN_INT_SET_0

Offset: 0x15c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH8_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_CH2_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_RXCIF_UNDERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.73 APBIF_APBIF_TXCIF_OVERRUN_INT_SET_0

Offset: 0x160 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH8_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_CH2_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_TXCIF_OVERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.74 APBIF_APBIF_APBDMA_UNDERRUN_INT_SET_0

Offset: 0x164 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	APBIF_CH8_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_CH2_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_APBDMA_TX_UNDERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.75 APBIF_APBIF_APBDMA_OVERRUN_INT_SET_0

Offset: 0x168 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9	0x0	APBIF_CH9_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
8	0x0	APBIF_CH8_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
7	0x0	APBIF_CH7_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
6	0x0	APBIF_CH6_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
5	0x0	APBIF_CH5_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
4	0x0	APBIF_CH4_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_CH2_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_CH1_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_CH0_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE

24.8.1.76 APBIF_AMX0_LIVE_STATUS_0

Offset: 0x178 │ Read/Write: RO │ Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23	X	AMX0_CH3_RX_ENABLED: 0 = DISABLE 1 = ENABLE
22	X	AMX0_CH2_RX_ENABLED: 0 = DISABLE 1 = ENABLE
21	X	AMX0_CH1_RX_ENABLED: 0 = DISABLE 1 = ENABLE
20	X	AMX0_CH0_RX_ENABLED: 0 = DISABLE 1 = ENABLE
16	X	AMX0_TX_ENABLED: 0 = DISABLE 1 = ENABLE
15	X	AMX0_CH3_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
14	X	AMX0_CH2_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
13	X	AMX0_CH1_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
12	X	AMX0_CH0_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	AMX0_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	AMX0_CH3_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	AMX0_CH2_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	AMX0_CH1_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	X	AMX0_CH0_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
0	X	AMX0_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

24.8.1.77 APBIF_ADX0_LIVE_STATUS_0

Offset: 0x180 │ Read/Write: RO │ Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23	X	ADX0_CH3_TX_ENABLED: 0 = DISABLE 1 = ENABLE
22	X	ADX0_CH2_TX_ENABLED: 0 = DISABLE 1 = ENABLE
21	X	ADX0_CH1_TX_ENABLED: 0 = DISABLE 1 = ENABLE
20	X	ADX0_CH0_TX_ENABLED: 0 = DISABLE 1 = ENABLE
16	X	ADX0_RX_ENABLED: 0 = DISABLE 1 = ENABLE
15	X	ADX0_CH3_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
14	X	ADX0_CH2_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
13	X	ADX0_CH1_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
12	X	ADX0_CH0_TX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
8	X	ADX0_RX_FIFO_FULL: 0 = DISABLE 1 = ENABLE
7	X	ADX0_CH3_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
6	X	ADX0_CH2_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
5	X	ADX0_CH1_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE
4	X	ADX0_CH0_TX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	X	ADX0_RX_FIFO_EMPTY: 0 = DISABLE 1 = ENABLE

24.8.1.78 APBIF_AMX_INT_MASK_0

Offset: 0x188 | Read/Write: R/W | Reset: 0x000000f1 (0bxxxxxxxxxxxxxxxxxxxxxxxxx1111xxx1)

Bit	Reset	Description
7	0x1	AMX0_CH3_RX_DONE: 0 = DISABLE 1 = ENABLE
6	0x1	AMX0_CH2_RX_DONE: 0 = DISABLE 1 = ENABLE
5	0x1	AMX0_CH1_RX_DONE: 0 = DISABLE 1 = ENABLE
4	0x1	AMX0_CH0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	0x1	AMX0_TX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.79 APBIF_ADX_INT_MASK_0

Offset: 0x190 | Read/Write: R/W | Reset: 0x000000f1 (0bxxxxxxxxxxxxxxxxxxxxxxxxx1111xxx1)

Bit	Reset	Description
7	0x1	ADX0_CH3_TX_DONE: 0 = DISABLE 1 = ENABLE
6	0x1	ADX0_CH2_TX_DONE: 0 = DISABLE 1 = ENABLE
5	0x1	ADX0_CH1_TX_DONE: 0 = DISABLE 1 = ENABLE
4	0x1	ADX0_CH0_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x1	ADX0_RX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.80 APBIF_AMX_INT_STATUS_0

Offset: 0x198 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000xxx0)

Bit	Reset	Description
7	0x0	AMX0_CH3_RX_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x0	AMX0_CH2_RX_DONE: 0 = DISABLE 1 = ENABLE
5	0x0	AMX0_CH1_RX_DONE: 0 = DISABLE 1 = ENABLE
4	0x0	AMX0_CH0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	AMX0_TX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.81 APBIF_ADX_INT_STATUS_0

Offset: 0x1a0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0000xxx0)

Bit	Reset	Description
7	0x0	ADX0_CH3_TX_DONE: 0 = DISABLE 1 = ENABLE
6	0x0	ADX0_CH2_TX_DONE: 0 = DISABLE 1 = ENABLE
5	0x0	ADX0_CH1_TX_DONE: 0 = DISABLE 1 = ENABLE
4	0x0	ADX0_CH0_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	ADX0_RX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.82 APBIF_AMX_INT_SOURCE_0

Offset: 0x1a8 │ Read/Write: RO │ Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7	X	AMX0_CH3_RX_DONE: 0 = DISABLE 1 = ENABLE
6	X	AMX0_CH2_RX_DONE: 0 = DISABLE 1 = ENABLE
5	X	AMX0_CH1_RX_DONE: 0 = DISABLE 1 = ENABLE
4	X	AMX0_CH0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	X	AMX0_TX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.83 APBIF_ADX_INT_SOURCE_0

Offset: 0x1b0 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7	X	ADX0_CH3_TX_DONE: 0 = DISABLE 1 = ENABLE
6	X	ADX0_CH2_TX_DONE: 0 = DISABLE 1 = ENABLE
5	X	ADX0_CH1_TX_DONE: 0 = DISABLE 1 = ENABLE
4	X	ADX0_CH0_TX_DONE: 0 = DISABLE 1 = ENABLE
0	X	ADX0_RX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.84 APBIF_AMX_INT_SET_0

Offset: 0x1b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000xxx0)

Bit	Reset	Description
7	0x0	AMX0_CH3_RX_DONE: 0 = DISABLE 1 = ENABLE
6	0x0	AMX0_CH2_RX_DONE: 0 = DISABLE 1 = ENABLE
5	0x0	AMX0_CH1_RX_DONE: 0 = DISABLE 1 = ENABLE
4	0x0	AMX0_CH0_RX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	AMX0_TX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.85 APBIF_ADX_INT_SET_0

Offset: 0x1c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000xxx0)

Bit	Reset	Description
7	0x0	ADX0_CH3_TX_DONE: 0 = DISABLE 1 = ENABLE
6	0x0	ADX0_CH2_TX_DONE: 0 = DISABLE 1 = ENABLE
5	0x0	ADX0_CH1_TX_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x0	ADX0_CH0_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	ADX0_RX_DONE: 0 = DISABLE 1 = ENABLE

24.8.1.86 APBIF_MISC_LIVE_STATUS0_0

Offset: 0x1d0 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12	X	I2S4_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
11	X	I2S3_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
10	X	I2S2_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
9	X	I2S1_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
8	X	I2S0_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
4	X	I2S4_CLKEN: 0 = DISABLE 1 = ENABLE
3	X	I2S3_CLKEN: 0 = DISABLE 1 = ENABLE
2	X	I2S2_CLKEN: 0 = DISABLE 1 = ENABLE
1	X	I2S1_CLKEN: 0 = DISABLE 1 = ENABLE
0	X	I2S0_CLKEN: 0 = DISABLE 1 = ENABLE

24.8.1.87 APBIF_MISC_LIVE_STATUS1_0

Offset: 0x1d4 | Read/Write: RO | Reset: 0x00000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	AMX0_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
8	X	AMX0_CLKEN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	X	ADX0_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
0	X	ADX0_CLKEN: 0 = DISABLE 1 = ENABLE

24.8.1.88 APBIF_MISC_LIVE_STATUS2_0

Offset: 0x1d8 | Read/Write: RO | Reset: 0x00000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10	X	DAM2_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
9	X	DAM1_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
8	X	DAM0_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
2	X	DAM2_CLKEN: 0 = DISABLE 1 = ENABLE
1	X	DAM1_CLKEN: 0 = DISABLE 1 = ENABLE
0	X	DAM0_CLKEN: 0 = DISABLE 1 = ENABLE

24.8.1.89 APBIF_MISC_LIVE_STATUS3_0

Offset: 0x1dc | Read/Write: RO | Reset: 0x00000X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	AUDIO_CFG_CLKEN: 0 = DISABLE 1 = ENABLE
8	X	APBIF_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
2	X	SPDIF_AUDIO_CLKEN: 0 = DISABLE 1 = ENABLE
1	X	SPDIF_OUT_CLKEN: 0 = DISABLE 1 = ENABLE
0	X	SPDIF_IN_CLKEN: 0 = DISABLE 1 = ENABLE

24.8.2 APBIF2 Registers

Audio channels 4 through 9 have their own set of 7 APBIF2 registers. Each channel has 32 bytes of address space. This subsection defines one complete set of APBIF2 registers. The register spaces per APBIF2 channel are listed in the table below.

Table 70: APBIF2 Channel Register Mapping

Register Space	Channel Registers
0x0 – 0x1f	Channel 4 APBIF2 Channel Registers
0x20 – 0x3f	Channel 5 APBIF2 Channel Registers
0x40 – 0x5f	Channel 6 APBIF2 Channel Registers
0x60 – 0x7f	Channel 7 APBIF2 Channel Registers
0x80 – 0x9f	Channel 8 APBIF2 Channel Registers
0xa0 – 0x17f	Channel 9 APBIF2 Channel Registers

24.8.2.1 APBIF2_CHANNELn_CTRL_0

There are 6 APBIF2 Channel Control Registers, one per channel (n = 4 through 9).

Offset: 0x0 + (n * 0x20) | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxx0000000000000000x000x000)

Bit	Reset	Description
31	0x0	TX_ENABLE: Enable TX channel to AUDIO. 0 = DISABLE 1 = ENABLE
30	0x0	RX_ENABLE: Enable RX channel to AUDIO. 0 = DISABLE 1 = ENABLE
29	0x0	LOOPBACK: Debug mode. The RX CIF is directly hooked up to the TX CIF. 0 = DISABLE 1 = ENABLE
23:16	0x0	TX_THRESHOLD. Trigger threshold level, number of free slots - 1. DMA request is triggered when the number of free slots is > TX_THRESHOLD. For example, If a DMA request needs to be generated when 4 free slots are available, this field needs to be set to 3.
15:8	0x0	RX_THRESHOLD: Trigger threshold level, # of data DWords - 1. DMA request is triggered when the number of DWords in the FIFO is > RX_THRESHOLD. For example, if a DMA request needs to be generated when 4 DWords are available, this field needs to be set to 3.
6	0x0	TX_PACK_EN: Enable packed data. 0 = DISABLE 1 = ENABLE
5:4	0x0	TX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16
2	0x0	RX_PACK_EN: Enable packed data. 0 = DISABLE 1 = ENABLE
1:0	0x0	RX_PACK: 0 = NOP 1 = RSVD 2 = PACK8_4 3 = PACK16

24.8.2.2 APBIF2_CHANNELn_CLEAR_0

There are 6 APBIF2 Channel Clear Registers, one per channel (n = 4 through 9).

Offset: 0x4 + (n * 0x20) | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	TX_SOFT_RESET: 0 = DISABLE 1 = ENABLE
30	0x0	RX_SOFT_RESET: 0 = DISABLE 1 = ENABLE

24.8.2.3 APBIF2_CHANNELn_STATUS_0

There are 6 APBIF2 Channel Status Registers, one per channel (n = 4 through 9).

Offset: 0x8 + (n * 0x20) | Read/Write: RO | Reset: 0xXXXX000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	TX_FREE_COUNT: Number of free slots in the TX FIFO.
23:16	X	RX_FREE_COUNT: Number of free slots in the RX FIFO.
1	X	TX_TRIG: TX FIFO threshold. 0 = DISABLE 1 = ENABLE
0	X	RX_TRIG: RX FIFO threshold. 0 = DISABLE 1 = ENABLE

24.8.2.4 APBIF2_CHANNELn_TXFIFO_0

There are 6 APBIF2 Channel TX FIFO Registers, one per channel (n = 4 through 9).

Offset: 0xc + (n * 0x20) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

24.8.2.5 APBIF2_CHANNELn_RXFIFO_0

There are 6 APBIF2 Channel RX FIFO Registers, one per channel (n = 4 through 9).

Offset: 0x10 + (n * 0x20) | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

24.8.2.6 APBIF2_AUDIOCIF_TXn_CTRL_0

There are 6 APBIF2 Audio CIF TX Control Registers, one per channel (n = 4 through 9).

When packed mode is enabled for a transmit channel, data is unpacked based on the APBIF2 TX channel CLIENT_BITS field.

- CLIENT_BITS = 0: Reserved. Do not use because 4 bit mode is not supported by the audio cluster.
- CLIENT_BITS = 1: (8 bit mode) : Four 8-bit samples are packed into one 32 bit word.
- CLIENT_BITS = 2: (12 bit mode): Two 12-bit samples are packed into one 32 bit word.

- CLIENT_BITS = 3: (16 bit mode): Two 16-bit samples are packed into one 32 bit word.
- CLIENT_BITS > 3: Packing is not possible.

Offset: 0x14 + (n * 0x20) | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD

Bit	R/W	Reset	Description
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.2.7 APBIF2_AUDIOCIF_RXn_CTRL_0

There are 6 APBIF2 Audio CIF RX Control Registers, one per channel (n = 4 through 9).

Offset: 0x18 + (n * 0x20) | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16

Bit	R/W	Reset	Description
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.3 Audio Registers

24.8.3.1 AUDIO_APBIF_RX0_0

The following macro defines a register for each RX port. No more than one field can be 1 in a register because an RX port can receive from only one TX port.

Example:

```
reg DAM0_RX0 incr4
0:0          DAM0_TX0          rw        i=0
1:1          I2S0_TX0          rw        i=0
2:2          I2S1_TX0          rw        i=0
;
```

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.2 AUDIO_APBIF_RX1_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.3 AUDIO_APBIF_RX2_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.4 AUDIO_APBIF_RX3_0

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

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.5 AUDIO_I2S0_RX0_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.6 AUDIO_I2S1_RX0_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.7 AUDIO_I2S2_RX0_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.8 AUDIO_I2S3_RX0_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.9 AUDIO_I2S4_RX0_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.10 AUDIO_DAM0_RX0_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.11 AUDIO_DAM0_RX1_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.12 AUDIO_DAM1_RX0_0

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.13 AUDIO_DAM1_RX1_0

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10	0x0	DAM1_TX0 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0 0 = DISABLE 1 = ENABLE

24.8.3.14 AUDIO_DAM2_RX0_0

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.15 AUDIO_DAM2_RX1_0

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.16 AUDIO_SPDIF_RX0_0

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.17 AUDIO_SPDIF_RX1_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.18 AUDIO_APBIF_RX4_0

Offset: 0x44 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.19 AUDIO_APBIF_RX5_0

Offset: 0x48 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.20 AUDIO_APBIF_RX6_0

Offset: 0x4c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.21 AUDIO_APBIF_RX7_0

Offset: 0x50 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.22 AUDIO_APBIF_RX8_0

Offset: 0x54 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.23 AUDIO_APBIF_RX9_0

Offset: 0x58 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.24 AUDIO_AMX0_RX0_0

Offset: 0x5c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.25 AUDIO_AMX0_RX1_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.26 AUDIO_AMX0_RX2_0

Offset: 0x64 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.27 AUDIO_AMX0_RX3_0

Offset: 0x68 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.3.28 AUDIO_ADX0_RX0_0

Offset: 0x6c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx00000000000000000000000)

Bit	Reset	Description
24	0x0	ADX0_TX3: 0 = DISABLE 1 = ENABLE
23	0x0	ADX0_TX2: 0 = DISABLE 1 = ENABLE
22	0x0	ADX0_TX1: 0 = DISABLE 1 = ENABLE
21	0x0	ADX0_TX0: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	AMX0_TX0: 0 = DISABLE 1 = ENABLE
19	0x0	APBIF_TX9: 0 = DISABLE 1 = ENABLE
18	0x0	APBIF_TX8: 0 = DISABLE 1 = ENABLE
17	0x0	APBIF_TX7: 0 = DISABLE 1 = ENABLE
16	0x0	APBIF_TX6: 0 = DISABLE 1 = ENABLE
15	0x0	APBIF_TX5: 0 = DISABLE 1 = ENABLE
14	0x0	APBIF_TX4: 0 = DISABLE 1 = ENABLE
13	0x0	SPDIF_TX1: 0 = DISABLE 1 = ENABLE
12	0x0	SPDIF_TX0: 0 = DISABLE 1 = ENABLE
11	0x0	DAM2_TX0: 0 = DISABLE 1 = ENABLE
10	0x0	DAM1_TX0: 0 = DISABLE 1 = ENABLE
9	0x0	DAM0_TX0: 0 = DISABLE 1 = ENABLE
8	0x0	I2S4_TX0: 0 = DISABLE 1 = ENABLE
7	0x0	I2S3_TX0: 0 = DISABLE 1 = ENABLE
6	0x0	I2S2_TX0: 0 = DISABLE 1 = ENABLE
5	0x0	I2S1_TX0: 0 = DISABLE 1 = ENABLE
4	0x0	I2S0_TX0: 0 = DISABLE 1 = ENABLE
3	0x0	APBIF_TX3: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	APBIF_TX2: 0 = DISABLE 1 = ENABLE
1	0x0	APBIF_TX1: 0 = DISABLE 1 = ENABLE
0	0x0	APBIF_TX0: 0 = DISABLE 1 = ENABLE

24.8.4 DAM Registers

24.8.4.1 DAM_CTRL_0

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxxxxxxxxxxxxxxx00000x00)

Bit	Reset	Description
31	DISABLE	SOFT_RESET: This bit is Auto Cleared. It resets the DAM logic including CIFs and only clears the DAM Enable & Channel Enables configuration bits. Software should wait until this bit is cleared. 0 = DISABLE 1 = ENABLE
7:4	X	FSOUT: Output DATA Sample Rate. 0 = FS8 1 = FS16 2 = FS44 3 = FS48
3	DISABLE	STEREO_MIXING_EN: This will enable stereo mixing. IMPORTANT: This requires DAM to be in bypass mode! Therefore FSOUT and FSIN must be equal for this to work. 0 = DISABLE 1 = ENABLE
1	DISABLE	CG_EN: Enables DAM second level clock gating 0 = DISABLE 1 = ENABLE
0	DISABLE	ENABLE: Enables the DAM 0 = DISABLE 1 = ENABLE

24.8.4.2 DAM_AUDIOCIF_OUT_CTRL_0

For the DAM output

The DAM output is fixed at 32 bits. CIF should convert the output to the desired format

CLIENT_BITS : BIT32

Offset: 0xc | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD

Bit	R/W	Reset	Description
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD

Bit	R/W	Reset	Description
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.4.3 DAM_CH0_CTRL_0

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx0000xxx00000000xxx0)

Bit	Reset	Description
18:16	0x0	FILT_STAGES: Number of filter stages minus 1 (excluding interpolation/decimation stages). For example for an 8->48 conversion, there are two IIR filter stages. Therefore 0x1 should be programmed. 8->48 : 1 8->44.1 : 2 16->48 : 0 16->44.1 : 3 44.1->16 : 2 44.1->8 : 2 48->16 : 0 48->8 : 1 0 = ONE 1 = TWO 2 = THREE 3 = FOUR 4 = FIVE 5 = SIX 6 = SEVEN 7 = EIGHT
11:8	X	FSIN: Input sample rate 0 = FS8 1 = FS16 2 = FS44 3 = FS48
7:4	0x0	DATA_SYNC: When a sample its ready in this channel, the bit says the mixer should wait for data in other channels. Each bit represents each channel, i.e., bit 1 -> CH 1. Bit 0 is meaningless for CH0. For CH0, 0b0000 is recommended since CH0 data should not wait for other channels.
0	DISABLE	ENABLE: 0 = DISABLE 1 = ENABLE

24.8.4.4 DAM_AUDIOCIF_CH0_CTRL_0

The DAM has 2 RX ports and 1 TX port.

Since LLF can handle only 16-bit mono data, CIF should be configured accordingly.

CLIENT_BITS : BIT16
CLIENT_CHANNELS : CH1
PACK : NOP

Offset: 0x1c | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD

Bit	R/W	Reset	Description
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD

Bit	R/W	Reset	Description
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.4.5 DAM_CH1_CTRL_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000010 (0bxxxxxxxxxxxxxxxxxxxxxxxxx0001xxx0)

Bit	Reset	Description
7:4	0x1	DATA_SYNC: When a sample is ready in this channel, the bit says the mixer should wait for data in other channels. Each bit represents each channel, i.e., bit 0 -> CH 0. Bit 1 is meaningless for CH1. For CH1, 0b0001 is recommended since CH1 data should wait for CH0 data.
0	DISABLE	ENABLE: 0 = DISABLE 1 = ENABLE

24.8.4.6 DAM_AUDIOCIF_CH1_CTRL_0

For the channel1 input

Offset: 0x2c | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7

Bit	R/W	Reset	Description
			7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.4.7 DAM_FARROW_PARAM_0

Offset: 0xf4 │ Read/Write: R/W │ Reset: 0xdee993a0 (0b11011110111010011001001110100000)

Bit	Reset	Description
31:16	0xdee9	STEP_DEN_RECIPROCAL: For sample rate conversions, 44.1 kHz to 16 kHz and 44.1 kHz to 8 kHz value should be 0xCCCD. For sample rate conversions, 16 kHz to 44.1 kHz and 8 kHz to 44.1 kHz value should be 0xDEE9.

Bit	Reset	Description
15:8	0x93	STEP_DEN: For sample rate conversions, 44.1 kHz to 16 kHz and 44.1 kHz to 8 kHz value should be 0xA0. For sample rate conversions, 16 kHz to 44.1 kHz and 8 kHz to 44.1 kHz value should be 0x93.
7:0	0xa0	STEP_NUM: For sample rate conversions, 44.1 kHz to 16 kHz and 44.1 kHz to 8 kHz value should be 0x93. For sample rate conversions, 16 kHz to 44.1 kHz and 8 kHz to 44.1 kHz value should be 0xA0.

24.8.5 I2S Registers

The I^2S Controller is designed to transport streaming audio-data between the system memory and an audio-codec. The controller supports I^2S format, Left Justified Mode format, Right Justified Mode format, and DSP mode format, as defined in the Philips inter-IC-sound (I^2S) bus specification.

The controller also supports the PCM and telephony (network) mode of data-transfer. Pulse-Code-Modulation (PCM) is a standard method used to digitize audio (particularly voice) patterns for transmission over digital communication channels.

The Telephony (network) mode is used to transmit and receive data to/from an external mono codec in a slot-based scheme of time-division multiplexing.

The controller has one transmit and one receive interface. The controller can be configured to operate either as a master or a slave.

24.8.5.1 I2S_CTRL_0

The I2S control register is used to configure bit formats (I2S, LJM, RJM, DSP, PCM, NW, TDM), bit sizes (8, 16, 20, 24 and 32), FIFO formats, Master/slave selection, LR polarity, and error interrupt enables.

I2S Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxxxxxxxx000x000xx00x000)

Bit	Reset	Description
31	0x0	XFER_EN_TX: Enable/disable I2S Transmit channel 0 = DISABLE 1 = ENABLE
30	0x0	XFER_EN_RX: Enable/disable I2S Receive channel 0 = DISABLE 1 = ENABLE
29	0x0	CG_EN: Enable/disable I2S second level clock gating 0 = DISABLE 1 = ENABLE
28	0x0	SOFT_RESET: This bit is auto-cleared. Resets I2S logic including CIFs and flow control. Configuration registers are not reset by soft reset. 0 = DISABLE 1 = ENABLE
27	0x0	TX_FLOWCTL_EN: Enable flow control in Transmit channel 0 = DISABLE 1 = ENABLE
14:12	0x0	FRAME_FORMAT: Frame format. 0: BASIC, LJM, RJM modes 1: DSP, PCM, NW, TDM modes 0 = LRCK_MODE 1 = FSYNC_MODE

Bit	Reset	Description
10	0x0	MASTER: Controller Master/Slave mode selection. 0: Do not drive LRCK and clock. 1: Drive LRCK and clock 0 = DISABLE 1 = ENABLE
9	0x0	LRCK_POLARITY: Left/Right Control Polarity. 0= Left channel when LRCK is low 1= Left channel when LRCK is high 0 = LOW 1 = HIGH
8	0x0	LPBK: Tx->Rx Loop Back Enable 0 = DISABLE 1 = ENABLE
5:4	0x0	BIT_CODE: sample compression/decompression 0: None 1: uLaw 2: aLaw 3: Reserved 0 = LINEAR 1 = uLAW 2 = ALAW 3 = RSVD
2:0	0x0	BIT_SIZE: Bit size. 0 = BIT_SIZE_RSVD 1 = BIT_SIZE_8 2 = BIT_SIZE_12 3 = BIT_SIZE_16 4 = BIT_SIZE_20 5 = BIT_SIZE_24 6 = BIT_SIZE_28 7 = BIT_SIZE_32

24.8.5.2 I2S_TIMING_0

The CHANNEL_BIT_CNT field of this register is used to program the number of bit-clocks per channel of LRCK (sampling rate) that the I^2S controller needs to send out in Master mode. This value is interpreted as follows:

- DSP mode: - Number of bit clocks (sclk) within (LEFT +RIGHT) channel width.
- Basic/LJM/RJM modes:- Number of bit clocks (sclk) within each individual channel width (LEFT or RIGHT).
- PCM/NW/TDM modes:- Number of bit clocks (sclk) per frame

The NON_SYM.EN can be used to program the I2S Controller to output a non-50:50 mark-space ratio on to the I2S bus. When the NON_SYM.EN Bit[12] is enabled, the Controller sends out exactly the programmed number of clock cycles on the left channel and one bit clock greater on the right channel. . This is used in Basic/LJM/RJM modes.

When NON_SYM.EN is enabled, the channel-bit-count should be programmed with the number of bit-clocks required in left-channel. This will ensure that the non-50:50 mark-space ratio is achieved with R_BCLK = L_BCLK + 1

The PCM/NW/TDM modes channel_bit_cnt can be calculated using the equation:

```
Channel_bit_cnt  = (frequency of bit_clk) / (2 * required sampling rate) -1;
```

If this calculation returns a fractional value, the non-symmetry feature of the controller should be used to attain the required sampling rate.

I2S Timing Register

Offset: 0x4 | Read/Write: R/W | Reset: 0x0000001f (0bxxxxxxxxxxxxxxxxxxxx0x00000011111)

Bit	Reset	Description
12	0x0	NON_SYM_EN: To enable non-symmetry mode. 0 = DISABLE 1 = ENABLE
10:0	0x1f	CHANNEL_BIT_CNT: I2S, LJM, RJM mode: Number of bit clocks in left or right channel. DSP mode: Number of bit clocks in LEFT+RIGHT channel TDM/NW/PCM mode: Number of bit clocks in the frame

24.8.5.3 I2S_OFFSET_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx00000000000xxxxx00000000000)

Bit	Reset	Description
26:16	0x0	RX_DATA_OFFSET: RX Data offset to fsync 0: No offset n: Data is offset by n bit clocks with respect to fsync
10:0	0x0	TX_DATA_OFFSET: TX Data offset to fsync 0: No offset n: Data is offset by n bit clocks with respect to fsync

24.8.5.4 I2S_CH_CTRL_0

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000xxxxxxxxxx00x000x000x000)

Bit	Reset	Description
31:24	0x0	FSYNC_WIDTH: Fsync width in terms of bit clocks 0: Fsync width is 1 clock wide. 1: Fsync width is 2 clocks wide. n: Fsync width is n+1 clocks wide.
13:12	0x0	HIGHZ_CTRL: HighZ control 0: Always drive SdataOut. 1: Tristate SdataOut if a slot is not valid. 2: Tristate SdataOut after driving last bit data for half a bitclk cycle instead of driving it for a full bit clock 0 = NOHIGHZ 1 = HIGHZ 2 = HIGHZ_ON_HALF_BIT_CLK
10	0x0	RX_BIT_ORDER: Configure to appear MSB or LSB first on sdata out 0: MSB first 1: LSB first 0 = MSB_FIRST 1 = LSB_FIRST
9	0x0	TX_BIT_ORDER: Configure to appear MSB or LSB first on sdata out 0: MSB first 1: LSB first 0 = MSB_FIRST 1 = LSB_FIRST

Bit	Reset	Description
8	0x0	EDGE_CTRL: Edge on which Data is driven. 0: Drive data on the negative edge and sample data on the positive edge 1: Drive data on the positive edge and sample data on the negative edge 0 = POS_EDGE 1 = NEG_EDGE
6:4	0x0	RX_MASK_BITS: Used for PCM mode 0 = ZERO 1 = ONE 2 = TWO 3 = THREE 4 = FOUR 5 = FIVE 6 = SIX 7 = SEVEN
2:0	0x0	TX_MASK_BITS: Used for PCM mode. Set these mask bits to get the exact bit size in PCM mode. For example, to get 13 bits of data, BIT_SIZE should be set to 3 (BIT16) and MASK should be set to 3 (THREE). 0 = ZERO 1 = ONE 2 = TWO 3 = THREE 4 = FOUR 5 = FIVE 6 = SIX 7 = SEVEN

24.8.5.5 I2S_SLOT_CTRL_0

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	TOTAL_SLOTS: Number of slots per fsync 0: Frame has 1 slot 1: Frame has 2 slots n: Frame has n+1 slots

24.8.5.6 I2S_AUDIOCIF_I2STX_CTRL_0

An I2S module has two AUDIOCIF sub-modules. In default, the first AUDIO port is for RX and the second AUDIO port is for TX. However, the direction can be changed with register programming so that I2S can have either two RX or two RX AUDIO ports.

CIF RX Port for I2S TX Path

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10

Bit	R/W	Reset	Description
			10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP

Bit	R/W	Reset	Description
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.5.7 I2S_AUDIOCIF_I2SRX_CTRL_0

CIF TX Port for I2S RX Path

Offset: 0x18 │ Read/Write: R/W │ Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24

Bit	R/W	Reset	Description
			6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.5.8 I2S_FLOWCTL_0

This register controls the flow control mechanism and is valid only with the TX mode. The flow control is only useful when the input stream to I^2S/SPDIF is already real-time spaced such as a live stream from Baseband through another I2S_RX. If the input stream is from DMA directly, the flow control should be disabled.

FLOWCTRL

Offset: 0x1c │ Read/Write: R/W │ Reset: 0x80000XXX (0b10xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	QUAD	FILTER: Use linear or quadratic filter. Tegra 4 devices support quadratic filter only. 0 = LINEAR 1 = QUAD
30	DISABLE	DUMMY_WR_EN: Insert a dummy frame when enabled 0 = DISABLE 1 = ENABLE
11:8	X	START_THRESHOLD: Flow control should wait for FIFO filled N-1, 0 means actual threshold = 1
7:4	X	HIGH_THRESHOLD: High-threshold for HIGH state N-1, 0 means actual threshold = 1
3:0	X	LOW_THRESHOLD: Low-threshold for LOW state N-1, 0 means actual threshold = 1

24.8.5.9 I2S_TX_STEP_0

Offset: 0x20 │ Read/Write: R/W │ Reset: 0x00008000 (0bxxxxxxxxxxxxxxxx1000000000000000)

Bit	Reset	Description
15:0	0x8000	STEP_SIZE: This is a compensation value for the clock difference between input and output. To catch up the clock difference quickly, this value is supposed to be bigger than clock

Bit	Reset	Description
		difference in terms of STEP_SIZE; however, to avoid sound quality degradation, it should not be much greater than the clock difference. For example, the input frame sampling rate is 44.095 kHz and the output frame sampling rate is 44.100 kHz, the clock error rate is (44100 - 44095) / 44100 = 0.000113. The step size used in the linear interpolation is 0x8000 (32768) step_size = 0.000113 * 32768 = 3.7. A threshold of 4 is recommended. If the value is less than 3 in the above example, the adjusted step size will not be able to catch up. The flow control FIFO will eventually become empty, leading to either NULL or replication of the last sample.

24.8.5.10 I2S_FLOW_STATUS_0

When the flow controller is used, the step size should be properly chosen depending on the clock frequency difference. The step size should be small enough not to degrade sound quality, but big enough to converge the FIFO depth.

If the step size is not big enough, the FIFO depth will either increase or decrease, which will cause FIFO overflow/underflow eventually. The flow controller monitor and counter can be used to fine-tune the step size. When the flow controller is enabled and a proper step size is chosen, the FIFO depth should be in [threshold -1 : threshold + 1].

When the flow controller monitor is enabled and the FIFO depth is deviated from the range, either FLOW_OVERFLOW or FLOW_UNDERFLOW will be set. Also, the interrupt signal will be asserted, if MONITOR_INT_EN is enabled. MONITOR_CLR clears the FLOW_OVERFLOW/UNDERFLOW as well as the interrupt signal to APBIF.

When the flow controller counter is enabled, it records how many samples are processed and how many times the step size is applied to either lower or raise the time step. In case of stereo samples, it counts as 1.

- FLOW_NORMAL - the total number of samples (only count left channel samples).
- FLOW_OVER - the number of samples with adding STEP_SIZE to the time step.
- FLOW_UNDER - the number of samples with subtracting STEP_SIZE from the time step.

Flow Controller Monitor/Counter

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x0000000X (0b00xxxxxxxxxxxxxxxxxxxxxxxxxx0xx00)

Bit	Reset	Description
31	NORMAL	FLOW_UNDERFLOW: FIFO depth is less than threshold - 1 0 = NORMAL 1 = UNDER
30	NORMAL	FLOW_OVERFLOW: FIFO depth is greater than threshold + 1 0 = NORMAL 1 = OVER
4	DISABLE	MONITOR_INT_EN: Enable monitor interrupt to APBIF 0 = DISABLE 1 = ENABLE
3	X	COUNTER_CLR: Clear counter
2	X	MONITOR_CLR: Clear monitor
1	DISABLE	COUNTER_EN: Enable counter 0 = DISABLE 1 = ENABLE
0	DISABLE	MONITOR_EN: Enable monitor 0 = DISABLE 1 = ENABLE

24.8.5.11 I2S_FLOW_TOTAL_0

Offset: 0x28 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNTER: the total number of left channel samples processed

24.8.5.12 I2S_FLOW_OVER_0

Offset: 0x2c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNTER: the number of +STEP_SIZEs

24.8.5.13 I2S_FLOW_UNDER_0

Offset: 0x30 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNTER: the number of -STEP_SIZEs

24.8.5.14 I2S_SLOT_CTRL2_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	RX_SLOT_ENABLES: Rx Slot enables. Used for TDM mode
15:0	0x0	TX_SLOT_ENABLES: Tx Slot enables. Used for TDM mode Only the enabled slots are used to transmit or receive data in the TDM mode.

24.8.6 SPDIF Registers

The S/PDIF consists of the following two major modules:

- The SPDIFOUT sub-module, which sends data to the "spdifout" port in IEC 60958-3 biphase-mark code format.
- The SPDIFIN sub-module, which retrieves data to the "spdifin" port in IEC 60958-3 biphase-mark code format.

24.8.6.1 SPDIF_CTRL_0

SPDIF Control Register

Note: Changing the state of TC_EN, TU_EN, LBK_EN, PACK, BIT_MODE while RX_EN and/or TX_EN and/or RX_BSY and/or TX_BSY is set can cause unexpected behavior and therefore shall not be attempted.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b000000xxxxxxxxxx00000xxx0xxxxxxx)

Bit	Reset	Description
31	0x0	FLOWCTL_EN: 1=Enable flow control 0 = DISABLE 1 = ENABLE
30	0x0	CAP_LC: 1=start capturing from left channel,0=start capturing from right channel. 0 = RIGHT_CH 1 = LEFT_CH
29	0x0	RX_EN: SPDIF receiver (RX): 1=enable, 0=disable. 0 = DISABLE 1 = ENABLE
28	0x0	TX_EN: SPDIF transmitter (TX): 1=enable, 0=disable. 0 = DISABLE 1 = ENABLE
27	0x0	TC_EN: Transmit Channel status: 1=enable, 0=disable. 0 = DISABLE 1 = ENABLE
26	0x0	TU_EN: Transmit user Data: 0 = DISABLE 1 = ENABLE
15	0x0	LBK_EN: Loopback test mode: 1=Enable internal loopback, 0=Normal mode. 0 = DISABLE 1 = ENABLE
14	0x0	PACK: Pack data mode: 1=Packeted left/right channel data into a single word, 0=Single data (16 bits need to be padded to match the interface data bit size) 0 = DISABLE 1 = ENABLE
13:12	0x0	BIT_MODE: 00=16-bit data. 01=20-bit data. 10=24-bit data. 11=raw data. This value should match ACIF CLIENT_BITS. 0 = MODE16BIT 1 = MODE20BIT 2 = MODE24BIT 3 = MODERAW
11	0x0	CG_EN: 1=Enable second level clock gating 0 = DISABLE 1 = ENABLE
7	0x0	SOFT_RESET: This bit is Auto Cleared. Resets I2S logic including CIFs and flow

Bit	Reset	Description
		control. Configuration registers are not reset by soft reset. 0 = DISABLE 1 = ENABLE

24.8.6.2 SPDIF_STROBE_CTRL_0

SPDIF Data Strobe Control Register

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00XX0000 (0bxxxxxxxxxxxxxxxx0xx00000xx000000)

Bit	R/W	Reset	Description
23:16	RO	X	PERIOD: Indicates the approximate number of detected SPDIFIN clocks within a biphase period.
15	RW	0x0	STROBE: SPDIFIN Data Strobe Mode 1=Manual-locked strobe 0=Auto-locked strobe (default)
12:8	RW	0x0	DATA_STROBES: Manual data strobe time within the biphase clock period (in terms of the number of oversampling clocks)
5:0	RW	0x0	CLOCK_PERIOD: Manual SPDIFIN biphase clock period (in terms of the number of over-sampling or 'spdifin' clocks)

24.8.6.3 SPDIF_AUDIOCIF_TXDATA_CTRL_0

An SPDIF module has four AUDIOCIF interfaces for TX data, RX data, TX user data, and RX user data. CIF RX port for SPDIF TX DATA

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11

Bit	R/W	Reset	Description
			11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.6.4 SPDIF_AUDIOCIF_RXDATA_CTRL_0

CIF TX port for SPDIF RX DATA

Offset: 0xc │ Read/Write: R/W │ Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4

Bit	R/W	Reset	Description
			4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF

Bit	R/W	Reset	Description
			1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.6.5 SPDIF_AUDIOCIF_TXUSER_CTRL_0

CIF RX Port for SPDIF USER Data

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4

Bit	R/W	Reset	Description
			1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.6.6 SPDIF_AUDIOCIF_RXUSER_CTRL_0

CIF TX Port for SPDIF USER Data

Offset: 0x14 │ Read/Write: R/W │ Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3

Bit	R/W	Reset	Description
			3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.6.7 SPDIF_CH_STA_RX_A_0

This 6-word receive channel data page buffer holds a block (192 frames) of channel status information. The order of receive is from LSB bit to MSB bit, and from CH_STA_RX_A to CH_STA_RX_F and back to CH_STA_RX_A.

Note: Only channel status bits from channel A (subframe 1) will be saved into this page buffer.

SPDIF Channel Status Rx Page Buffer Register

Offset: 0x18 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C31_RX_C0: Channel status bits [31:0]; one bit per audio sample

24.8.6.8 SPDIF_CH_STA_RX_B_0

Offset: 0x1c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C63_RX_C32: Channel status bits [63:32]; one bit per audio sample

24.8.6.9 SPDIF_CH_STA_RX_C_0

Offset: 0x20 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C95_RX_C64: Channel status bits [95:64]; one bit per audio sample

24.8.6.10 SPDIF_CH_STA_RX_D_0

Offset: 0x24 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C127_RX_C96: Channel status bits [127:96]; one bit per audio sample

24.8.6.11 SPDIF_CH_STA_RX_E_0

Offset: 0x28 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C159_RX_C128: Channel status bits [159:128]; one bit per audio sample

24.8.6.12 SPDIF_CH_STA_RX_F_0

Offset: 0x2c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_C191_RX_C160: Channel status bits [191:160]; one bit per audio sample

24.8.6.13 SPDIF_CH_STA_TX_A_0

This 6-word transmit channel data page buffer holds a block (192 frames) of channel status information. The order of transmission is from LSB bit to MSB bit, and from CH_STA_TX_A to CH_STA_TX_F and back to CH_STA_TX_A.

Note: The channel status data from this page buffer will be used only when PACK=1, or when BIT_MODE=00/01/10.
Each channel status bit will be sent out to "both" subframes.

SPDIF Channel Status Tx Page Buffer Register

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C31_TX_C0: Channel status bits [31:0]; one bit per audio sample

24.8.6.14 SPDIF_CH_STA_TX_B_0

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C63_TX_C32: Channel status bits [63:32]; one bit per audio sample

24.8.6.15 SPDIF_CH_STA_TX_C_0

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C95_TX_C64: Channel status bits [95:64]; one bit per audio sample

24.8.6.16 SPDIF_CH_STA_TX_D_0

Offset: 0x3c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C127_TX_C96: Channel status bits [127:96]; one bit per audio sample

24.8.6.17 SPDIF_CH_STA_TX_E_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_C159_TX_C128: Channel status bits [159:128]; one bit per audio sample

24.8.6.18 SPDIF_CH_STA_TX_F_0

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

Bit	Reset	Description
31:0	0x0	TX_C191_TX_C160: Channel status bits [191:160]; one bit per audio sample

24.8.6.19 SPDIF_FLOWCTL_CTRL_0

This register controls the flow control mechanism and is valid only with the TX mode. The flow control is only useful when the input stream to I2S/SPDIF is already real-time spaced such as a live stream from Baseband through another I2S_RX. If the input stream is from DMA directly, the flow control should be disabled.

Offset: 0x70 | Read/Write: R/W | Reset: 0x80000XXX (0b10xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	QUAD	FILTER: to use linear or quadratic filter. Tegra 4 devices support quadratic filter only. 0 = LINEAR 1 = QUAD
30	DISABLE	DUMMY_WR_EN: Insert a dummy frame when enabled 0 = DISABLE 1 = ENABLE
11:8	X	START_THRESHOLD: Flow control should wait for FIFO filled N-1, 0 means actual

Bit	Reset	Description
		threshold 1
7:4	X	HIGH_THRESHOLD: High-threshold for HIGH state N-1, 0 means actual threshold 1
3:0	X	LOW_THRESHOLD: Low-threshold for LOW state N-1, 0 means actual threshold 1

24.8.6.20 SPDIF_TX_STEP_0

Offset: 0x74 │ Read/Write: R/W │ Reset: 0x00008000 (0bxxxxxxxxxxxxxxxx1000000000000000)

Bit	Reset	Description
15:0	0x8000	STEP_SIZE: This is a compensation value for the clock difference between input and output. To catch up the clock difference quickly, this value is supposed to be bigger than clock difference in terms of STEP_SIZE, but to avoid the sound quality degradation it should not be much greater than the clock difference. For example, the input frame sampling rate is 44.095 kHz, the output frame sampling rate is 44.100 kHz, and the clock error rate is (44100 - 44095) / 44100 = 0.00011.3 The step size used in the linear interpolation is 0x8000 (32768) step_size = 0.000113 * 32768 = 3.7. A threshold of 4 is recommended. If the value is less than 3 in the above example, the adjusted step size will not be able to catch up. The flow control FIFO will eventually become empty, leading to either NULL or replication of the last sample.

24.8.6.21 SPDIF_FLOW_STATUS_0

When the flow controller is used, the step size should be properly chosen depending on the clock frequency difference. The step size should be small enough not to degrade sound quality, but big enough to converge the FIFO depth. If the step size is not big enough, the FIFO depth will either increase or decrease, which will cause FIFO overflow/underflow eventually.

The flow controller monitor and counter can be used to fine-tune the step size. When the flow controller is enabled and a proper step size is chosen, the FIFO depth should be in [threshold -1 : threshold + 1].

When the flow controller monitor is enabled and the FIFO depth is deviated from the range, either FLOW_OVERFLOW or FLOW_UNDERFLOW will be set. Also, the interrupt signal will be asserted, if MONITOR_INT_EN is enabled. MONITOR_CLR clears the FLOW_OVERFLOW/UNDERFLOW as well as the interrupt signal to APBIF.

When the flow controller counter is enabled, it records how many samples are processed and how many times the step size is applied to either lower or raise the time step. In case of stereo samples, it counts as 1.

- FLOW_NORMAL - the total number of samples (only count left channel samples).
- FLOW_OVER - the number of samples with adding STEP_SIZE to the time step.
- FLOW_UNDER - the number of samples with subtracting STEP_SIZE from the time step.

Flow Controller Monitor/Counter

Offset: 0x78 │ Read/Write: R/W │ Reset: 0x0000000X (0b00xxxxxxxxxxxxxxxxxxxxxxxxxx0xx00)

Bit	Reset	Description
31	NORMAL	FLOW_UNDERFLOW: FIFO depth is less than threshold - 1 0 = NORMAL 1 = UNDER
30	NORMAL	FLOW_OVERFLOW: FIFO depth is greater than threshold + 1 0 = NORMAL 1 = OVER
4	DISABLE	MONITOR_INT_EN: enable monitor interrupt to APBIF 0 = DISABLE 1 = ENABLE
3	X	COUNTER_CLR: clear counter

Bit	Reset	Description
2	X	MONITOR_CLR: clear monitor
1	DISABLE	COUNTER_EN: enable counter 0 = DISABLE 1 = ENABLE
0	DISABLE	MONITOR_EN: enable monitor 0 = DISABLE 1 = ENABLE

24.8.6.22 SPDIF_FLOW_TOTAL_0

Offset: 0x7c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNTER: the total number of left channel samples processed

24.8.6.23 SPDIF_FLOW_OVER_0

Offset: 0x80 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNTER: the number of +STEP_SIZEs

24.8.6.24 SPDIF_FLOW_UNDER_0

Offset: 0x84 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNTER: the number of -STEP_SIZEs

24.8.7 AMX Registers

24.8.7.1 AMX_CTRL_0

AMX Control Register

This register specifies the input stream parameters (the number of audio channels in the input stream and number of bits per sample).

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxx0000xxxxxxxx)

Bit	Reset	Description
31	0x0	SOFT_RESET: This bit is auto cleared. Resets AMX logic. 0 = DISABLE 1 = ENABLE
30	0x0	CG_EN: Second level clock gating enable. 0 = DISABLE 1 = ENABLE
11:10	0x0	MSTR_CH_NUM: Designated master channel. 0 = CH0 1 = CH1 2 = CH2 3 = CH3
9:8	0x0	CH_DEP: This field determines whether to wait for all enabled channels to have data before a transfer or to start sending when any one of the input channels has data. 0x0: Send output when ALL enabled channels have data. 0x1: Send output when any one of the enabled channels has data. 0x2: Send output when designated master stream has data. 0x3: Reserved 0 = WT_ON_ALL 1 = WT_ON_ANY 2 = WT_ON_MASTER 3 = RSVD

24.8.7.2 AMX_IN_CH_CTRL_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxx0000)

Bit	Reset	Description
11	0x0	CH3_FORCE_DISABLE: Channel 3 enable 0 = DISABLE 1 = ENABLE
10	0x0	CH2_FORCE_DISABLE: Channel 2 enable 0 = DISABLE 1 = ENABLE
9	0x0	CH1_FORCE_DISABLE: Channel 1 enable 0 = DISABLE 1 = ENABLE
8	0x0	CH0_FORCE_DISABLE: Channel 0 enable 0 = DISABLE 1 = ENABLE
3	0x0	CH3_EN: Channel 3 enable 0 = DISABLE 1 = ENABLE
2	0x0	CH2_EN: Channel 2 enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	CH1_EN: Channel 1 enable 0 = DISABLE 1 = ENABLE
0	0x0	CH0_EN: Channel 0 enable 0 = DISABLE 1 = ENABLE

24.8.7.3 AMX_OUT_BYTE_EN0_0

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

Bit	Reset	Description
31:0	0x0	BYTE_EN: Byte enables for bytes 0 to 31

24.8.7.4 AMX_OUT_BYTE_EN1_0

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

Bit	Reset	Description
31:0	0x0	BYTE_EN: Byte enables for bytes 32 to 63

24.8.7.5 AMX_AUDIORAMCTL_AMX_CTRL_0

Offset: 0x10 | Read/Write: R/W | Reset: 0xX0000000 (0bxxxxxxxxxxxxxxxxxx000xxxxx0000000)

Bit	R/W	Reset	Description
31	RO	X	READ_BUSY: 0 = DONE 1 = BUSY
14	RW	0x0	RW: 0 = READ 1 = WRITE
13	RW	0x0	RESET_HW_ADR: 0 = DONE 1 = BUSY
12	RW	0x0	HW_ADR_EN: 0 = DISABLE 1 = ENABLE
6:0	RW	0x0	RAM_ADR

24.8.7.6 AMX_AUDIORAMCTL_AMX_DATA_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

24.8.7.7 AMX_AUDIOCIF_OUT_CTRL_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD

Bit	R/W	Reset	Description
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE

Bit	R/W	Reset	Description
			1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.7.8 AMX_AUDIOCIF_CH0_CTRL_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS:

Bit	R/W	Reset	Description
			0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.7.9 AMX_AUDIOCIF_CH1_CTRL_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5

Bit	R/W	Reset	Description
			5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.7.10 AMX_AUDIOCIF_CH2_CTRL_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS:

Bit	R/W	Reset	Description
			0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.7.11 AMX_AUDIOCIF_CH3_CTRL_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12

Bit	R/W	Reset	Description
			3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.8 ADX Registers

24.8.8.1 ADX_CTRL_0

ADX Control Register

This register specifies the input stream parameters (the number of audio channels in the input stream and number of bits per sample).

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	SOFT_RESET: This bit is auto cleared. Resets ADX logic 0 = DISABLE 1 = ENABLE
30	0x0	CG_EN: Second-level clock gating enable. 0 = DISABLE 1 = ENABLE

24.8.8.2 ADX_OUT_CH_CTRL_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000xxxx0000)

Bit	Reset	Description
11	0x0	CH3_FORCE_DISABLE: Ch 3 enable 0 = DISABLE 1 = ENABLE
10	0x0	CH2_FORCE_DISABLE: Ch 2 enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	0x0	CH1_FORCE_DISABLE: Ch 1 enable 0 = DISABLE 1 = ENABLE
8	0x0	CH0_FORCE_DISABLE: Ch 0 enable 0 = DISABLE 1 = ENABLE
3	0x0	CH3_EN: Ch3 enable 0 = DISABLE 1 = ENABLE
2	0x0	CH2_EN: Ch2 enable 0 = DISABLE 1 = ENABLE
1	0x0	CH1_EN: Ch1 enable 0 = DISABLE 1 = ENABLE
0	0x0	CH0_EN: Ch0 enable 0 = DISABLE 1 = ENABLE

24.8.8.3 ADX_IN_BYTE_EN0_0

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

Bit	Reset	Description
31:0	0x0	BYTE_EN: Byte enables for bytes 0 to 31

24.8.8.4 ADX_IN_BYTE_EN1_0

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

Bit	Reset	Description
31:0	0x0	BYTE_EN: Byte enables for bytes 32 to 63

24.8.8.5 ADX_AUDIORAMCTL_ADX_CTRL_0

Offset: 0x10 | Read/Write: R/W | Reset: 0xX0000000 (0bxxxxxxxxxxxxxxxxxx000xxxxx0000000)

Bit	R/W	Reset	Description
31	RO	X	READ_BUSY: 0 = DONE 1 = BUSY
14	RW	0x0	RW: 0 = READ 1 = WRITE
13	RW	0x0	RESET_HW_ADR: 0 = DONE 1 = BUSY
12	RW	0x0	HW_ADR_EN: 0 = DISABLE 1 = ENABLE
6:0	RW	0x0	RAM_ADR

24.8.8.6 ADX_AUDIORAMCTL_ADX_DATA_0

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

24.8.8.7 ADX_AUDIOCIF_IN_CTRL_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32

Bit	R/W	Reset	Description
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.8.8 ADX_AUDIOCIF_CH0_CTRL_0

Offset: 0x1c | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16

Bit	R/W	Reset	Description
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.8.9 ADX_AUDIOCIF_CH1_CTRL_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15

Bit	R/W	Reset	Description
			15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.8.10 ADX_AUDIOCIF_CH2_CTRL_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG

Bit	R/W	Reset	Description
			3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

24.8.8.11 ADX_AUDIOCIF_CH3_CTRL_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000110X (0b00000000xxxx0000x001x00100000x00)

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
27:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
19:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8 8 = CH9 9 = CH10 10 = CH11 11 = CH12 12 = CH13 13 = CH14 14 = CH15 15 = CH16
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32

Bit	R/W	Reset	Description
10:8	RW	0x1	CLIENT_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24 6 = BIT28 7 = BIT32
7:6	RW	0x0	EXPAND: 0 = ZERO 1 = ONE 2 = LFSR 3 = RSVD
5:4	RW	0x0	STEREO_CONV: 0 = CH0 1 = CH1 2 = AVG 3 = RSVD
3	RW	0x0	REPLICATE: 0 = DISABLE 1 = ENABLE
2	RO	X	DIRECTION: 0 = TXCIF 1 = RXCIF
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

25.0 DISPLAY CONTROLLER

The Tegra® 4 architecture has two entirely independent display controllers. They are intended to support two independent display devices, typically a local display panel and an external HDMI TV. Other configurations are possible such as two local panels. Each display controller can run at a different clock rate and drive a different resolution panel.

The Tegra 4 display controllers are significantly more efficient in the way they fetch from memory compared with prior Tegra devices. They use line buffers to store scan lines required for re-use, to avoid redundant fetches when scaling or using a tiled surface format.

25.1 Features

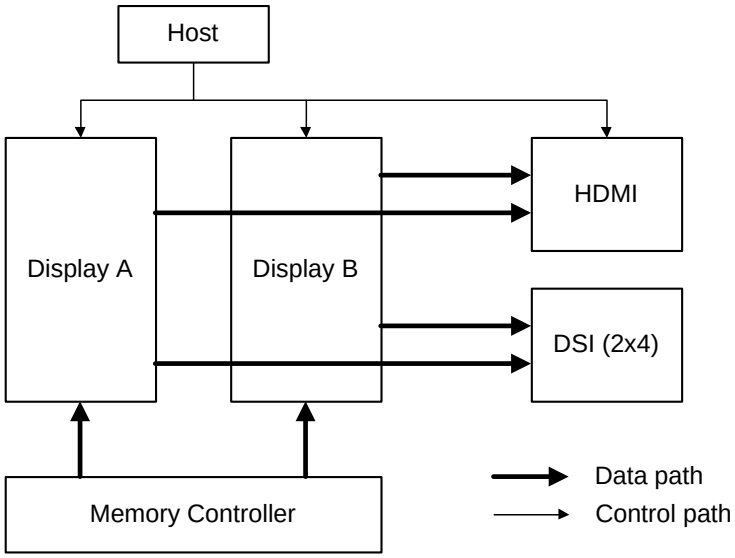
Specific enhancements include:

- Use of line buffering to avoid refetching data from memory on subsequent lines
- Efficient fetch for scaling
- Support for 32-byte memory atoms
- Improved handling of display buffer underflow
- Support for high-quality scaling/filtering across all windows
- Support up to Quad-HD resolution
- Support for Image Rotation in hardware
- Extending the cursor size to 256x256 with full color and alpha-blending
- Display Color Management unit (based on a LUT and CSC in the output path)

Features removed:

- Analog TV Out
- LCD parallel interface

Figure 85: Display Controller Top-Level Block Diagram



25.1.1 Output Capabilities

- 1 or 2 captive panels on two 4-lane DSI channels, operating either independently for 1 or 2 panels, or ganged together as 2x4 for a single panel.
  - Maximum panel size of 2560x1600 @ 60 Hz using 2x 4-lane ganged DSI channels
  - Independent resolution and pixel clock when not ganged
- 1 External Display
  - On HDMI. Maximum 1920x1080p @ 60 Hz (stereo or not)
  - On HDMI. Maximum 4096x2160p @ 30 Hz
  - With HDCP and multi-channel audio

25.1.2 Color Management Unit (CMU)

- Gamma (tone curve) conversion
- Color gamut conversion
- Allow conversion of sRGB content for a non-sRGB panel, for predictable colors (color matching)
- Enhanced PRISM display for these panels, as the PRISM algorithm assumes sRGB gamma
- Enable TCON and panels with non sRGB response
- Allows color calibrated displays, and enables end-to-end Camera / ISP -> Display color quality

25.1.3 Window A/B/C

- Data Formats
  - B4G4R4A4
  - B5G5R5A, AB5G5R5, B5G6R5
  - B8G8R8A8, R8G8B8A8
  - YCbCr420P , YCbCr422P, YCbCr422RP (planar format)
  - YCbCr422 (packed format)
  - Palletized format limited to 8-bit mode.
- Three 256x8-bit RAMs (color palette) used for gamma correction.
- Horizontal and Vertical Flip
- Horizontal and Vertical Scaling
  - Horizontal scale-down by up to 4x, subject to clock speed limitations
  - Vertical scale-down by up to 4x for 2Bpp or 2x for 4Bpp, again subject to clock speed limitations
  - Scale-up by up to 4000x
- Filtering
  - 6-tap, 16-phase programmable H filter
  - 2-tap, 16-phase programmable V filter
- YCbCr to RGB color space converter
- Digital Vibrance (8-level)

The table below describes a breakdown of features for a Tegra 4 Display on a window basis.

Head	Maximum Resolution	Window	Scaling	Planar Rotation	Packed Rotation
A INTERNAL	2560x1600	A	**Y**	**Y**	**Y**
		B	**Y**	**N**	**Y**
		C	**Y**	**N**	**Y**
B EXTERNAL	Quad-HD (4096x2160)	A	**Y**	**N**	**N**
		B	**Y**	**N**	**N**
		C	**Y**	**N**	**N**

25.2 Block Diagrams

Figure 86: Display Controller Front End Block Diagram

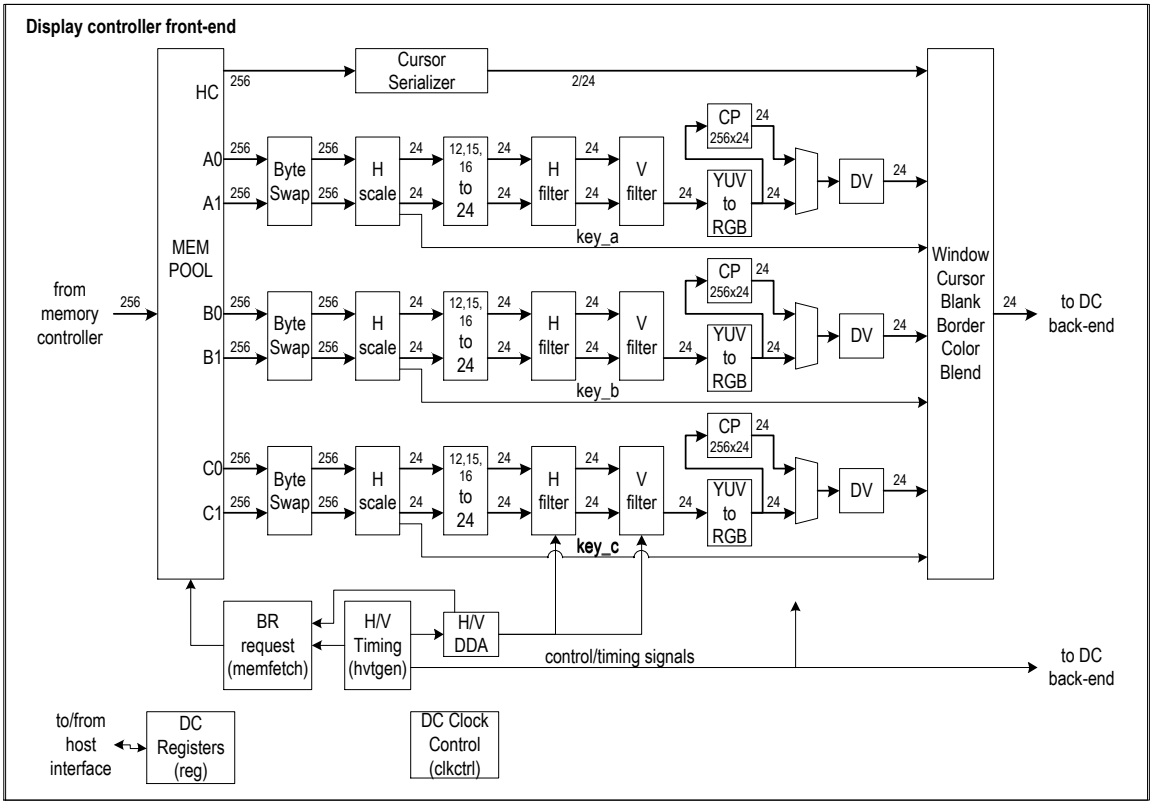
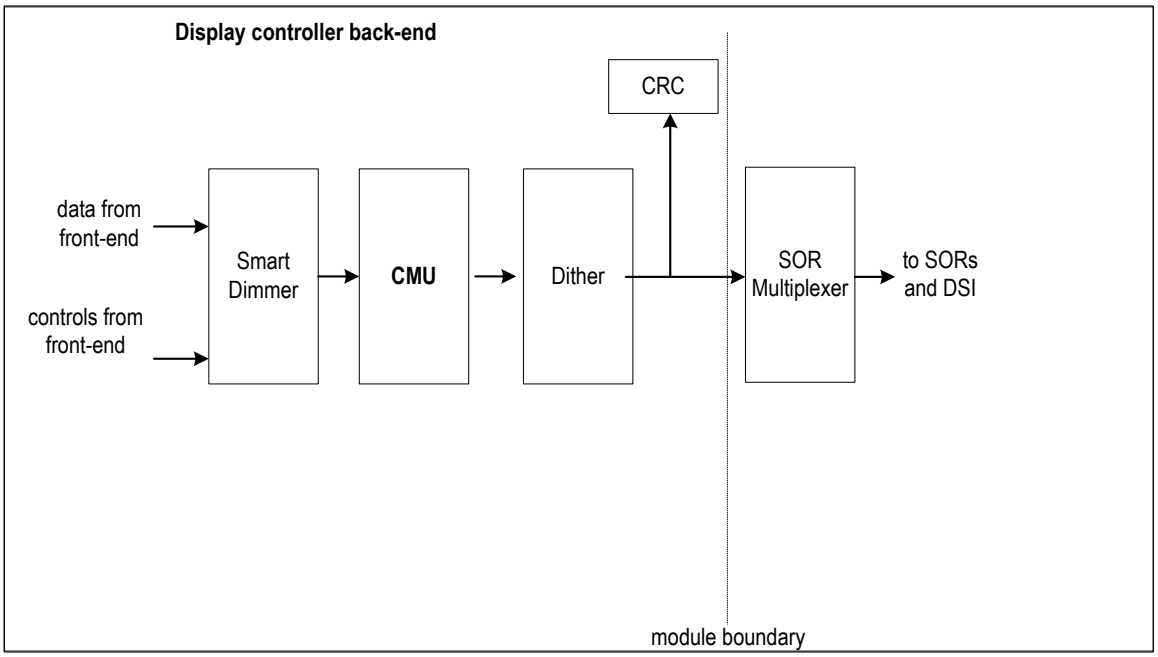


Figure 87: Display Controller Back End Block Diagram



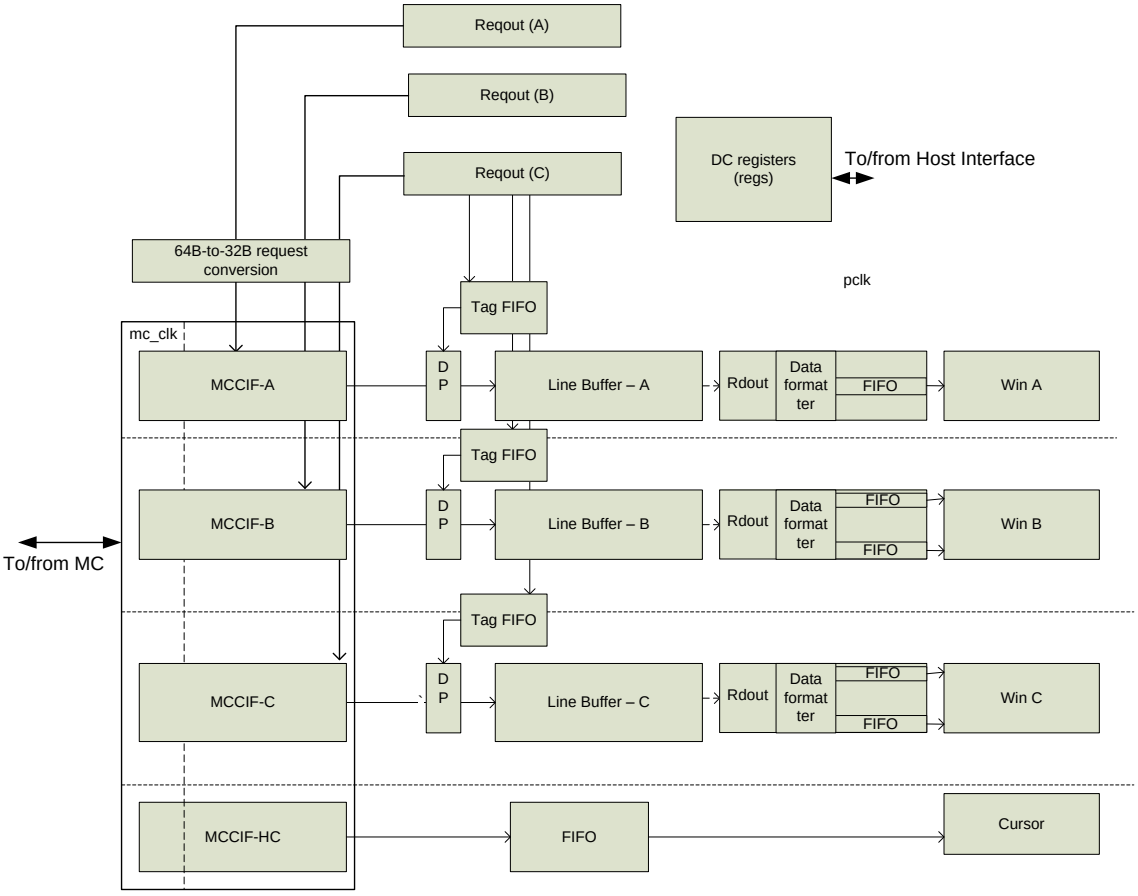
25.3 Display Controller Description

25.3.1 MEMFETCH

The top-level block diagram of the display memory interface for Tegra 4 devices is given in Figure 88. The diagram shows 3 memory fetch engines associated with the 3 windows in the display controller. Each window memory fetch engine has a line buffer and MCCIF.

Requests are issued to the memory and data is returned via MCCIF to the window line buffer. Data from the line buffer is subsequently read out, unpacked and passed to the window scaler and filter units. The cursor fetch engine does not contain line buffers.

Figure 88: Memfetch Block Diagram



25.3.1.1 Request Engine

The request engine is responsible for fetching the window image from memory. Logic to translate the X, Y coordinates to a linear memory address is used to make 32B requests to memory. This logic previously resided in the MCCIF.

A pulse signal is sent to the request engine from the display timing generator at the beginning of the vertical blanking period to alert the request engine to start requesting the image data. Tagged 32B requests are sent to the MCCIF. The thread ID specifying the line location of the return data is passed to the tag FIFO between the request and response engines. The thread ID is read by the data packer block when the return data is received and the data is steered to the appropriate line(s) in the line buffer.

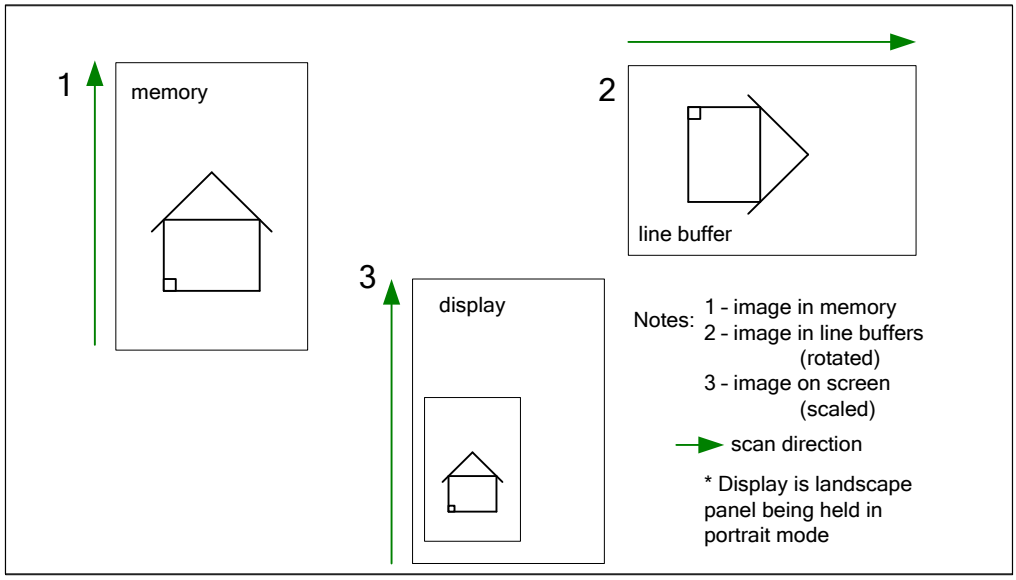
The request engine must handle the following surface formats:

- Pitched linear surface
- Tiled surface – 16B x 16 line
  - 16B x 2 line – subset with incrementing X (non-rotated)
  - 16B x 2 line – subset with incrementing Y (rotated surface)

Rotation Support

Previous versions of display support 0° or 180° rotation or image flip. Tablet use of portrait mode has now made rotation (90° or 270°) a common use-case. To support rotation the memfetch unit will rotate the data at the input to the line buffers. Put simply memfetch will walk the Y axis and store these columns in memory as rows to the display. The figure below illustrates the image orientation and shape at different stages in the pipeline for rotation.

Figure 89: Example Rotation Use-Case



Memfetch only supports rotation in tiled mode. This is due to the amount of horizontal pixel data read in pitched linear versus tiled format. For pitched linear 32B of horizontal data is fetched compared to tiled format where only 16B of horizontal data is fetched using the 16B x 4 line block linear mode. Since rotation switches rows and columns the 4 horizontal pixels become 4 rows in the line buffer. Therefore the minimum number of lines when rotation is enabled is 4. The number of lines grows to 5 when scaling is enabled.

The request for 16Bx4 tiled format must be communicated to MCCIF via addressing bits 6 and 7. Memfetch will increment the Y instead of X coordinate for subsequent accesses and after reaching maximum Y will reset the coordinates to (X+16B, 0) for the next fetch of columns from memory.

An additional bit (SCAN_COLUMN) is defined to indicate if the image needs to be scanned out row by row or column by column. When this bit is enabled display need to scan out lines column by column. This will provide the required 90°/270° rotated images. To achieve 90° or 270° rotation, software must use a combination of H_DIR, V_DIR and SCAN_COLUMN as shown in the table below. It illustrates how the image is scanned in to display by the memfetch unit for each combination of (SCAN_COLUMN, H_DIR, V_DIR).

SCAN_COLUMN	H_DIR	V_DIR	Orientation
0	0	0	0 degrees
0	0	1	N/A – vertical flip
0	1	0	N/A – horizontal flip
0	1	1	180 degrees
1	0	0	N/A – vertical flip followed by rotation
1	0	1	90 degrees
1	1	0	N/A – horizontal flip followed by rotation
1	1	1	270 degrees

The window dimension registers start to get confusing when rotation is introduced. Therefore it's important to note which registers apply to pre-rotation and which to post rotation.

The following registers apply to pre-rotation:

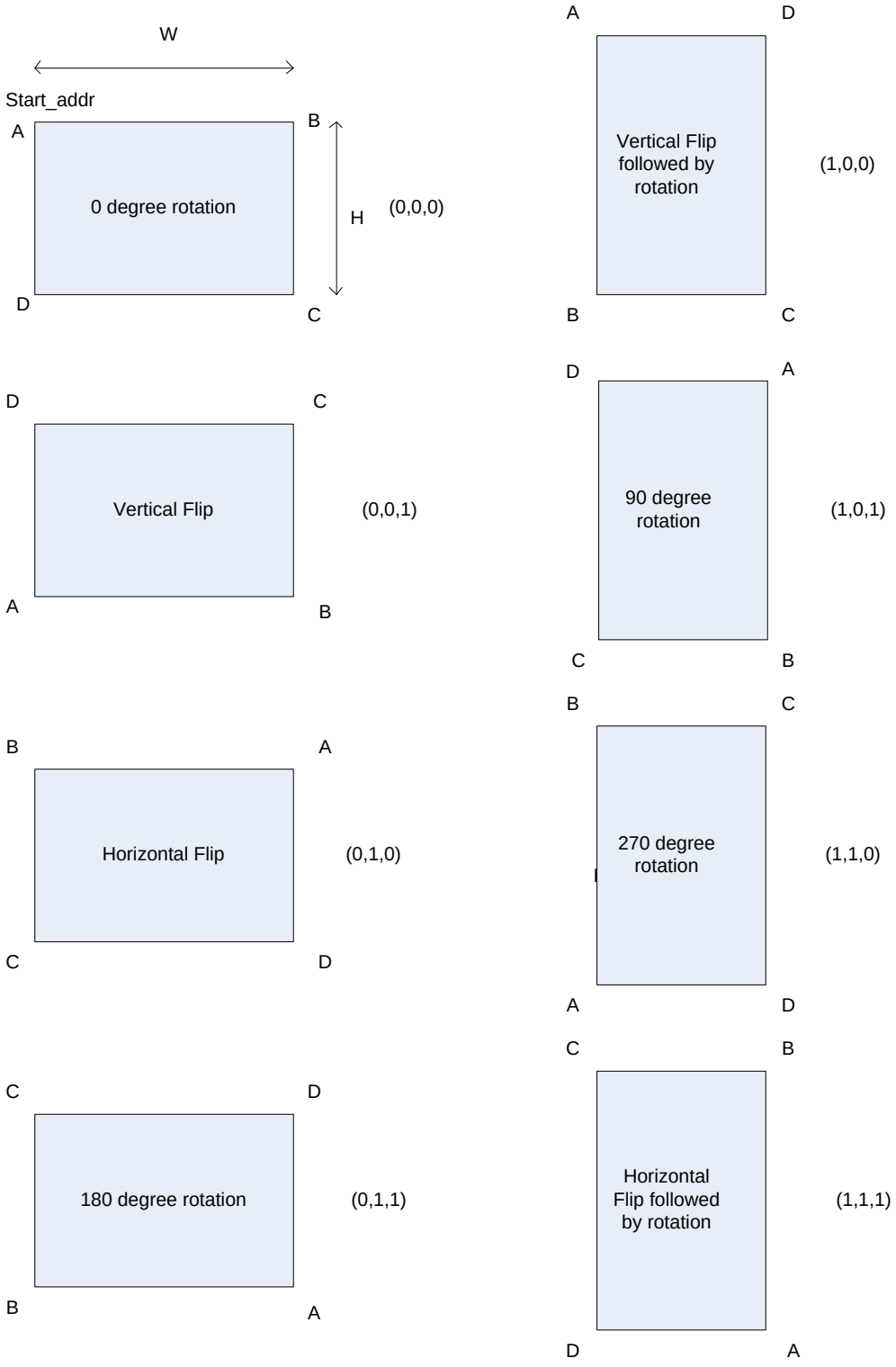
- B_WIN_OPTIONS
  - H_DIR
  - V_DIR
  - SCAN_COLUMN
- B_LINE_STRIDE
- B_START_ADDR_*
- B_ ADDR_H_OFFSET_*
- B_ ADDR_V_OFFSET_*

The following registers apply to post rotation:

- B_WIN_OPTIONS
  - H_FILTER*
  - V_FILTER*
  - SCAN_COLUMN
- B_PRESCALED_SIZE
- B_SIZE
- B*_INITIAL_DDA
- B_DDA_INCREMENT

Because the memfetch unit is essentially walking the Y axis in memory the majority of the window registers and its associated logic remain unchanged. Therefore except for B_ADDR_H/V_OFFSET horizontal and vertical directions are with respect to post rotation.

Figure 90: Rotation Combinations



25.3.1.2 Line Buffer

The memfetch line buffer allows the display controller to retain one or more scan line of window data locally so that no further reading of the scan line is required from memory. The scenarios where this is required are:

- Vertical scaling and filtering
  - Requires 3 scan lines of data for non-rotated memory fetches
  - Requires 5 scan lines of data for rotated memory fetches of packed data
- Reading tiled data
  - Data is sent in 2 scan lines for non-rotated memory fetches
  - Data is sent in 4 scan lines for rotated memory fetches

Data is received from the memory interface and steered to the proper location in the line buffer. Data is read out of the line buffer and unpacked based on the pixel format. These pixels are sent to the window scaler logic in the display pipe.

In the vertical scaling and filtering scenario, up to five scan lines of data are required for the vertical scaler engine for it to generate the proper scaled output. In the tiled mode scenario, only a single scan-line is output but the 32B returned data contains portions of two scan lines. To ensure adequate storage of RGB packed data format and steady bandwidth consumption to the display controller line buffers will be deep enough to hold up to five lines of maximum width of packed pixel data based on the maximum use-case supported by the given window.

Support for YUV planar rotation when reading data in 16Bx2 format requires a larger amount of data storage. Each plane of data will return 16B per color component. Due to scaling it is required to keep a scan line of data in the buffer when fetching the next block of 16. This will require an additional byte of data. Therefore the buffer requirements for YUV planar are:

- Number of bytes per plane = 17B
  - Minimum data fetch size = 16B
  - Scaling requires additional scan-line remain
  - Pixel component depth per plane = 1B
- Number of planes = 3
- Line buffer width = 2560 pixels – Tegra 4 maximum width
- Line buffer size = 17 bytes/plane* 3 planes/pixel * 2560 pixels per line
  - 130560 bytes

Since the line buffer storage requirement for YUV planar is close to three times (3X) the RGB packed format, the Tegra 4 display only supports buffering support for YUV planar rotation on 1 window. This restriction applies to YUV422 packed data as well due to the implementation of expanding the data to YUV444 in the line buffer. The other two windows will contain buffering to support non YUV planar rotation (51200 bytes).

Below are the equations used to calculate buffer requirements for a given rotated video image.

- YUV444 (packed):
  - Scaling – 5 lines*4Bpp*width
  - No scaling – 4 lines*4Bpp*width
- YUV422 (packed):
  - Scaling – 9 lines*4Bpp(as data is expanded)*width
  - No scaling – 8 lines*4Bpp(as data is expanded)*width
- YUV420 (planar):
  - Scaling: 3 (number of surfaces)*17*width (17 lines @ 1B/line)
  - No scaling: 3 (number of surfaces)*16*width (16 lines @ 1B/line)

As previously stated the purpose of the line buffer is to prevent re-fetch of data from memory. The line buffer size requirement to prevent re-fetch depends on the use-case. Based on the use case and method of packing/unpacking pixels stored in the line buffer there will be an inherent pixel latency associated with the buffer. The memory controller can take advantage of this latency to temporarily stop sending pixel data to display for cases such as changing the memory clock frequency.

Note that the line buffer can be written or read either with single or multiple lines per clock. For non-rotation use cases, line buffers can be loaded either one or two lines at a time corresponding to pitched linear or tiled data formats. For rotation use cases, line buffers load up to 16 lines at a time based on pixel format. Line buffers are read one or two lines at a time based on vertical scaling. The following tables and figures illustrate the line buffer size as well as inherent latency for various use cases for packed ARGB formats (4Bpp). For rotation use-case only tiled formats are supported. Tiled format is 16Bx16 lines.

Table 71: Non-Rotation Use Cases

Scenario	Vertical Scaling Enabled	Data Format	Line buffer size required to avoid re-fetch	Pixel Latency of Line Buffer
A	NO	Pitched-linear	0	0
B	NO	Tiled	1 line	0 - 1 line
C	YES	Pitched-linear	1 line	0
D	YES	Tiled	3 lines	1 line

Table 72: Rotation Use Cases

Scenario	Vertical Scaling Enabled	Data Format	Line buffer size required to avoid re-fetch	Pixel Latency of Line Buffer
A	NO	Tiled	4 line	1 - 4 lines
B	YES	Tiled	5 lines	2 - 5 lines
C	NO	YUV Planer	16 line	1 - 16 lines
D	YES	YUV Planer	17 lines	2 - 17 lines

The pixel latency refers to the number of lines available in the line buffer when the line buffer is full between when a given pixel is written to and read from the line buffer. For some of these use cases, there is a relatively long period of time after a pixel is written to the buffer before it is needed in the scan line. However it is important to note that in every case where a scan line or more of data exists the scan lines are being written at a rate of ½ or ¼ of the pixel clock rate since either 2 or 4 scan lines are being read at once from memory at a rate of 1 pixel per clock.

The fixed size of the Tegra 4 display buffers is given below. Each buffer size has a corresponding “maximum use-case” scenario from those listed above. For those scenarios that are not “maximum use-case”, the portion of the line buffer beyond what the scenario required can be used as latency buffering. Indeed, even the “maximum use-case” scenario can use a portion of the pixel latency as latency buffering.

Table 73: Tegra 4 Display Line Buffer Sizes

Line Buffer Size (KB)	Maximum Use-Case	Maximum Resolution
128	Rotation – Scenario C	2560x1600
50	Rotation – Scenario B	2560x1600
48	Non-Rotation – Scenario D	Quad-HD (4096x2160)

To take advantage of the portion of the line buffer dedicated for handling latency MEMFETCH will send flags to MC indicating when a programmable threshold is crossed. For each scenario, software must calculate the portion of the line buffer that can be used for latency buffering and set the threshold accordingly. Below are is a list of the flags MEMFETCH will generate:

- SPOOLUP
  - Signal name: csr_display0a2mc_spool_up
  - Active from when data is requested during Vblank until active space
  - Active space starts with 1st pixel of window sent to display pipe
- DVFS buffering

- Requires HWM threshold
- Signal name: csr_display0a2mc_ready_for_latency_event
- Active when:
    - Buffer exceeds HWM threshold
    - Window is disabled
    - From the time all data is read for current frame until data is requested for subsequent frame
- Deactivated with occurrence of underflow for the remainder of the frame

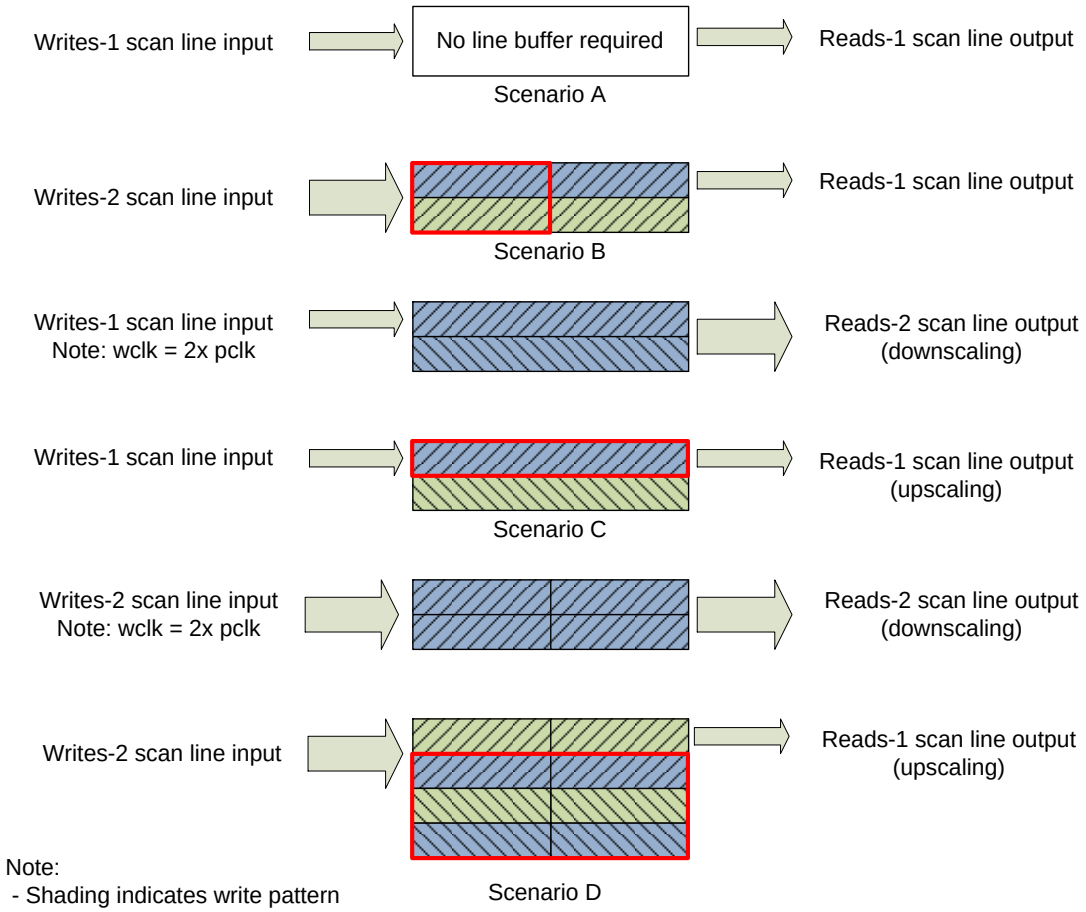
- LATENCY ALLOWANCE SCALE FACTOR buffering
    - Requires HWM/LWM thresholds
    - Signal names:
        - csr_display0a2mc_la_th_above_hi = level > HWM
            - deactivate with occurrence of underflow for remainder of frame
            - activate when window is disabled
        - csr_display0a2mc_la_th_below_lo = level < LWM
            - activate with occurrence of underflow for remainder of frame
            - deactivate when window is disabled

The programmable threshold will simply keep a running count of the number of bytes in the buffer and compare that value against the one in the threshold register.

Line buffering for planer YUV requires separate storage per plane. Each line is read by sending 1KB of Y requests followed by IKB of U then 1KB of V requests. This is maintained until the entire line of YUV is read. Subsequent line requests are not issued for any component until all component requests have been issued for the current line. MEMFETCH does allow for changing the weights of the given requests per plane to help control the behavior of the requests per plane.

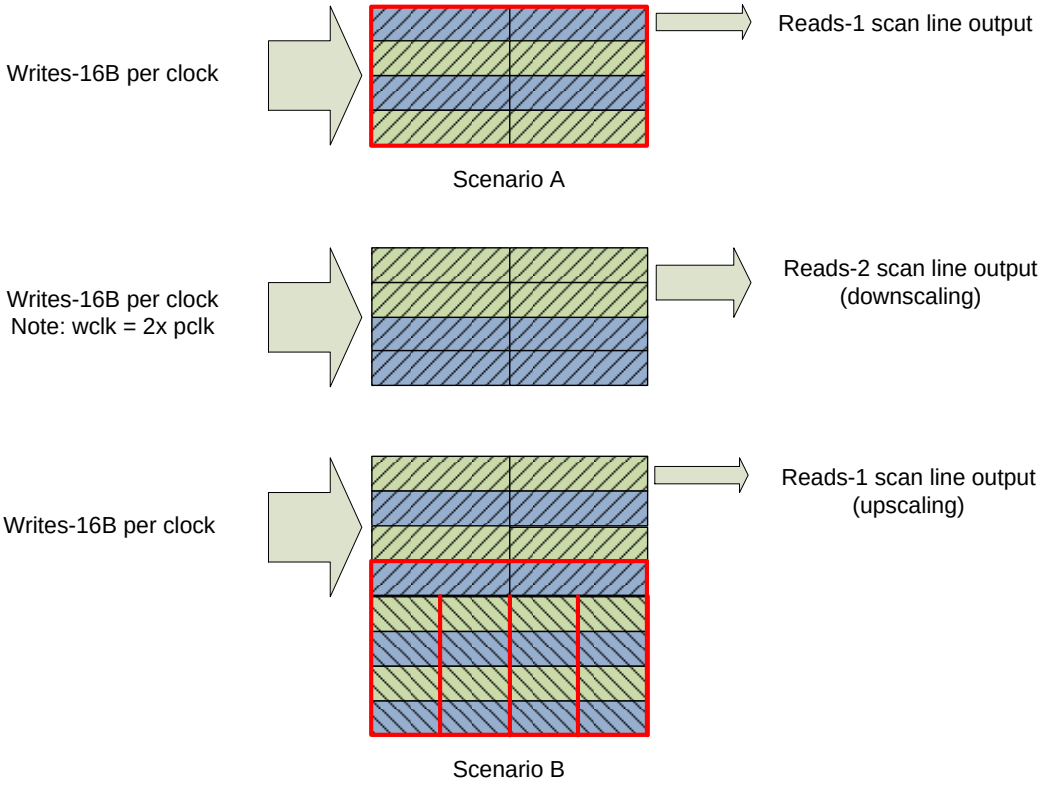
For non-rotated YUV planer use cases, the size of the line buffer is less than that of 4B ARGB since the pixel format is only 1.5B/pixel. However, rotated planer YUV requires storage of 16B for each line per plane. Contact your NVIDIA AE should you need further support on this.

Figure 91: Non-Rotation Line Buffer Use Cases



Note:
- Shading indicates write pattern
- Color indicates read pattern
- Red box indicates line buffer max size to avoid re-fetch

Figure 92: Rotation Line Buffer Use Cases



Note:
- Shading indicates write pattern
- Color indicates read pattern
- Red boxes indicate line buffer size needed to avoid refetch
- Red lines indicate progress of filling line buffer as scan lines are read

25.3.1.3 Handling of Underflow

Display is an isochronous client that requires a steady bandwidth allocation. If data is not returned at a fairly constant rate and the line buffers run dry then data will not appear on the screen at the proper time. Eventually when data does start to flow to the display controller the image will appear shifted and there is no way for the image data to catch up to where it should be on the screen. This undesirable lack of guaranteed data is called underflow and it can happen in 2 forms:

- Short term underflow – pixels worth of missing data
- Long term underflow – scan-lines worth of missing data

For short term underflow it is possible for the display controller to catch up to the correct pixel by the start of the next scan-line by reading pixels during the horizontal blanking period. Short term underflow is being managed by Tegra display controller in this manner. For long term underflow it would be impossible for the display controller to throw out that much data. Short term underflow may arise when the memory controller is over-taxed based on a momentary surge in memory client requests. Long term underflow may happen when memory controller frequency changes are being made.

Tegra 4 display windows will address short term underflow similar to what is done in Tegra 3. Long term underflow is also handled in a manner similar to Tegra 3 during the active frame portion. Below is a description of Tegra 4 Underflow recovery improvements:

- Underflow recovery within a frame

- Similar to Tegra 3:
    - Start counter of underflow pixels when underflow occurs
    - Last valid pixel is held at the beginning at the display pipeline for duration of the underflow
    - Underflow pixels that eventually return to display are stored in line buffer and immediately read/discarded
    - Operation continues (even during HBlank) until all underflow pixels have been discarded
    - Once valid pixel is present at display pipe input memfetch operations return to normal mode
- Limit requests issued at the end of a frame if window is in underflow using the following method:
    - For cases when underflow continues to the end of the frame
    - Add threshold counter (B_UFLOW_THRESHOLD)
    - Track prescaled scanout
    - Stop issuing frame requests when
        - prescaled scanout + threshold is > prescaled size

- Underflow recovery between frames
    - Add a FIFO before DP logic to determine which of the responses that are coming back are for the previous frame
    - Drop pixels from previous frame at 64B/pclk rate
    - Within display pipe, reset the underflowed pixel counters
    - Limit requests issued at the end of a frame if window is in underflow using method described above

The Tegra 4 display handles cursor underflow in the following manner:

- Interface between memfetch and display pipeline will sense when underflow occurs.
- When underflow occurs display pipeline will drive cursor alpha value of 0 for the remainder of the underflow period. This could be until the end of 1 or more scan-lines.
- Memfetch will not drive valid to display pipeline until the start of the scan-line when data is available.
- Display pipeline will not assert ready to memfetch until the beginning of the scan-line when data is available and memfetch is driving valid.
- Memfetch keeps track of number of clocks of missing data and when data does appear it reads out the data until the end of the scan line is reached.

It is important to highlight underflows for debug purposes. Current display units raise an interrupt to indicate that underflow has occurred. The Tegra 4 display continues to drive an interrupt when an underflow is detected. To further assist with highlighting underflows memfetch drives a solid color from the window that experiences an underflow if the associated enable bit is active.

25.3.2 Surface/Color Formats

In general, the Tegra 4 display supports the same pixel and surface formats used in Tegra 3. These pixel formats are not supported by the Tegra 4 display controller:

- P1 – 1bpp palletized
- P2 – 2bpp palletized
- P4 – 4bpp palletized
- RGB666
- YUV422RA

Below is a list of the surface and pixel formats supported by the Tegra 4 display controller. Formats are lsb-to-msb.

Surface formats:

- Pitched linear – scan-line data
- Tiled – 16B x 16 line

Pixel formats:

- RGB packed
  - P8 – 8bpp palletized
  - B4G4R4A4 - 16bpp
  - B5G5R5A- 16bpp
  - AB5G5R5 - 16bpp
  - B5G6R5 - 16bpp
  - B8G8R8A8- 32bpp
  - R8G8B8A8- 32bpp
- YUV packed
  - YCbCr422 – packed CbYCrYCbYCrY…
  - YUV422 – like YCbCr422 but with “True” YUV
- YUV planar
  - YCbCr422P (3 surfaces)
  - YCbCr422R (3 surfaces, rotated - U and V are same horizontal width as Y, but decimated by 2 vertically. 422R differs from 422P by decimating horizontally and replicating vertically)
  - YCbCr420P (3 surfaces)
  - YUV422P – like YCbCr422P but with “True” YUV
  - YUV422R – like YCbCr422R but with “True” YUV
  - YUV420P – like YCbCr420P but with “True” YUV

True YUV formats are supported in display along with YCrCb. YUV differs from YCbCr in that UV is signed -127 to 128 where CbCr is offset 128 and ranges 0 to 255.

25.3.3 Horizontal Filter

Memfetch and scalar provide 6 adjacent pixels per clock, along with DDA fraction indicating filter kernel phase. Fraction selects among 16 sets of coefficients for filter kernel.

25.3.4 Vertical Filter

When enabled, memfetch provides 2 adjacent lines in parallel, which are independently horizontally filtered before being sent to vertical filter. The fractional DDA from vertical scalar (indicating filter phase) selects among 16 filter kernel coefficients. For coefficient entry N, the weights are N and 1-N.

25.3.5 Color Space Converter

Most of the display datapath is RGB. The color space conversion unit can convert YCbCr and YUV to RGB. It can also do some RGB to RGB conversions.

25.3.6 Color Palette/LUT

To support gamma and palletized formats, the LUT has three 256x8-bit tables, one each for red, green, blue. The three tables are independent and can represent arbitrary functions. For 4-bit palette modes, only the bottom 16 entries are used.

25.3.7 Digital Vibrance

Using the same algorithm as NVIDIA® GeForce GPUs, Digital Vibrance can increase the saturation of colors.

25.3.8 Cursor

Cursor supports two pixel formats:

- Legacy 2-bit-per-pixel, where the four values are background color, foreground color, transparent, and invert. Foreground and background color are programmable via registers with 24-bit RGB values. Transparent means cursor is invisible and desktop shows through. Invert means desktop RGB is inverted (red' = 255 – red, etc.). Pixels are packed 4 per byte, and each line is 16 bytes pitch, regardless of whether it is 32 or 64 pixels wide.
  - Cursor size of 32x32 or 64x64
- 32-bpp RGBA pitch linear
  - Full color with alpha
  - Cursor size of:

    32x32

    64x64

    128x128

    256x256

The blending required for alpha cursor will occur after the blending stage and use the following blend equation:

Final_image = cursor_alpha * cursor_color + (1-cursor_alpha) * blending_stage_output_color

Note: Legacy cursor does not support per-pixel or global alpha.

Cursor can be clipped to either the display, or window A, B, or C. Thus the cursor active region for blending must be computed according to that mode. In regions where cursor is clipped, or outside the cursor entirely, blending behaves as if cursor is disabled (i.e., previous blend stage passes through.)

Note: The Tegra 4 display cursor supports partial clipping but does not support trivial reject of the cursor when the cursor is completely outside of the window region. It is the responsibility of software to ensure that the cursor is at least partially visible.

25.3.8.1 Legacy Support for Blending

The Tegra 4 display supports cursor modes used in previous generations of the Tegra display. Legacy cursor blending uses 2 bits per pixel and programming for this mode will remain unchanged.

25.3.8.2 32-bpp RGBA Cursor Mode

In 32-bpp RGBA cursor mode, legacy adaption is disabled. Cursor RGBA is blended with the last stage of the window blending unit. Cursor alpha value controls this blending.

25.3.9 Blending

The Tegra 4 display uses weighted blending between windows. Legacy cursor blending must be disabled when full-color alpha-blended cursor is in use.

The Tegra 4 device uses "parallel" blending.

Each Tegra 4 display controller has three windows (called A, B, and C) and a cursor. The three windows and cursor are combined to form the final output image; this operation is called "blending" or "compositing".

Each window and cursor has an independent position and size on the output raster. Areas with no window are filled with a background (“border”) color, which is programmable. The border color, however, does not participate in blending, except for cursor. In other words, the background under windows is effectively opaque black.

The cursor is always on top of the three windows and is logically a separate blending stage. The three windows do not have separate sequential stages or order; they are essentially peers and all contribute color in parallel to the final result.

In the example image below, three windows plus cursor are shown. The cursor is the upper-right square. The red area is “border” color. As you can see, the border color does not show within a window, but may show within the cursor. There are three areas where two windows overlap, and an area where all three overlap.

Figure 93: Window Blending Example

At each pixel of the output raster, the three windows can overlap in eight different ways:

- A only
- B only
- C only
- A and B
- A and C
- B and C
- A and B and C
- No window

The “no window” case is just filled with border color, as mentioned above. The seven other overlap cases each have separate blending equations. For example, Window A has four overlap conditions:

- A only
- A and B
- A and C
- A and B and C

These can be generalized for any window as:

- One window: self only
- Two windows: self + “other window 1”
- Two windows: self + “other window 2”
- Three windows: self plus other two windows

The blending control registers are organized per-window as the four cases above. Thus there are 12 sets of blending registers: three windows times four overlap cases each. There are additional cases to cover color-key match and non-match, but this document does not address color key.

The fundamental blending equation is

$$Color_{OUT} = Weight_A * Color_A + Weight_B * Color_B + Weight_C * Color_C$$

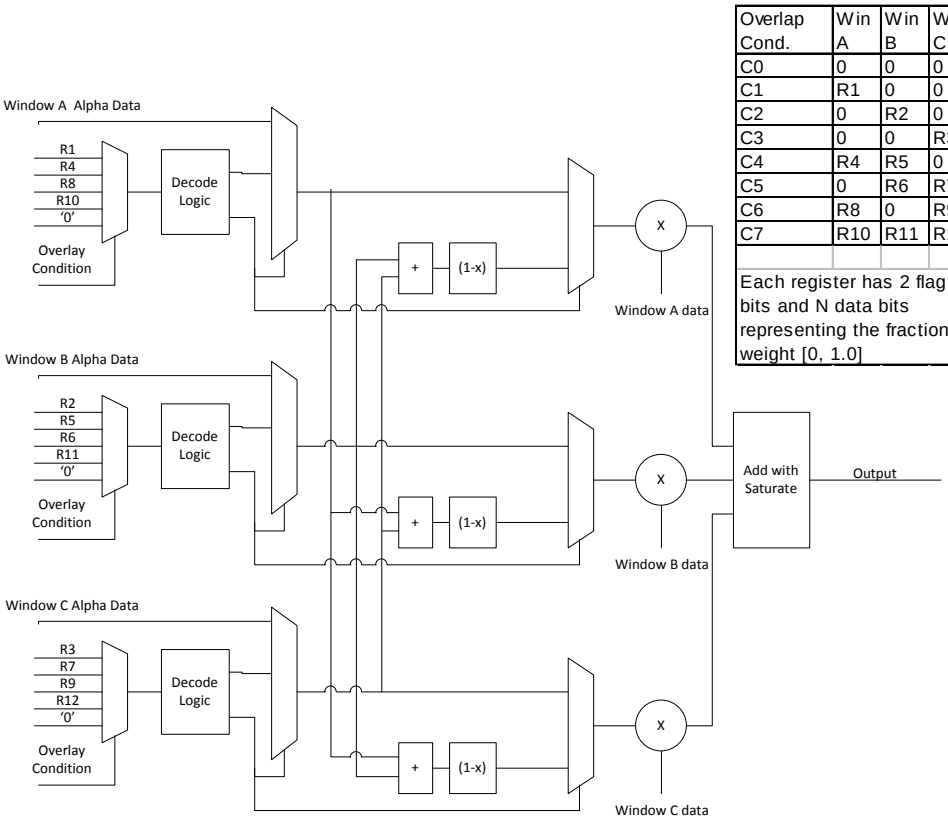
Where:

- Weight for each window is specified by blending control registers.
- Color is the result of each windows’ processing pipeline.

If a window does not participate in an overlap case, its weight is 0.

This architecture is shown schematically in the figure below.

Figure 94: Blending Schematic



Overlap Cond.	Win A	Win B	Win C
C0	0	0	0
C1	R1	0	0
C2	0	R2	0
C3	0	0	R3
C4	R4	R5	0
C5	0	R6	R7
C6	R8	0	R9
C7	R10	R11	R12

Each register has 2 flag bits and N data bits representing the fractional weight [0, 1.0]

For each of the four cases, each involved window has the following blending parameters:

- BLEND_CONTROL: FIX_WEIGHT, ALPHA_WEIGHT, DEPENDENT_WEIGHT, or PREMULT_WEIGHT
- BLEND_WEIGHT. 8-bit weight used for FIX_WEIGHT case

BLEND_CONTROL specifies the windows' weight for the blending equation:

- FIX_WEIGHT: the weight is from BLEND_WEIGHT register. This allows a window without per-pixel alpha to have a constant alpha.
- ALPHA_WEIGHT: The window surfaces' per-pixel alpha is used. The pixel RGB has not been pre-multiplied by alpha.
- DEPENDENT_WEIGHT: The window weight is the inverse of the *other* window(s)' weight(s). For example, in a two-window overlap case, if the other window has per-pixel alpha, then this window's weight is 255 – other_window_alpha. In a three-window overlap case, the weight is 255 – other_window1_weight – other_window2_weight.
- PREMULT_WEIGHT: Like ALPHA_WEIGHT except the pixel RGB has been pre-multiplied by its alpha.

25.3.9.1 Color Key

When color key is enabled on a window, there is an additional blending mode. When the key color is not matched, the window is forced to the foreground (i.e., the other two windows' blending weights are forced to 0.) When the color key is matched, blending proceeds normally. The window with color key would be programmed with FIX_WEIGHT and BLEND_WEIGHT=0 for all overlap cases, so that in the color key match case, the key color is not visible.

25.3.9.2 Two-Window Blending

A over B, where B is an opaque bottom window, can be achieved with:

- A BLEND_CONTROL = PREMULT_WEIGHT
- B BLEND_CONTROL = DEPENDENT_WEIGHT

Giving $Color_{OUT} = Color_A + (1 - Alpha_A) * Color_B$.

25.3.9.3 Three-Window Blending

The parallel blending described above is different than Porter-Duff style blending. Two-window Porter-Duff style "A over B" blending can be done. Three-window Porter-Duff-style blending – A over (B over C) – can be done in some but not all cases.

The three window case -- A over (B over C) – uses the general equation:

$$Color_{OUT} = Color_A + (1 - Alpha_A) * Color_B + (1 - Alpha_A)(1 - Alpha_B) * Color_C$$

In cases where two windows (for example, B and C) are opaque, this simplifies to:

$$Color_{OUT} = Color_A + (1 - Alpha_A) * Color_B + 0 * Color_C$$

which can be supported.

There are some limitations in the case where two of three windows are not opaque and all overlap. In Tegra 4 devices, a BLEND_CONTROL can be set to PREMULT_WEIGHT for the $Color_A$ term but the proper $Color_B$ or $Color_C$ terms cannot be achieved. Both B and C require inverse of window A weight but the blender only allows one window to be DEPENDENT_WEIGHT. Also, it cannot supply $(1 - Alpha_A)(1 - Alpha_B)$.

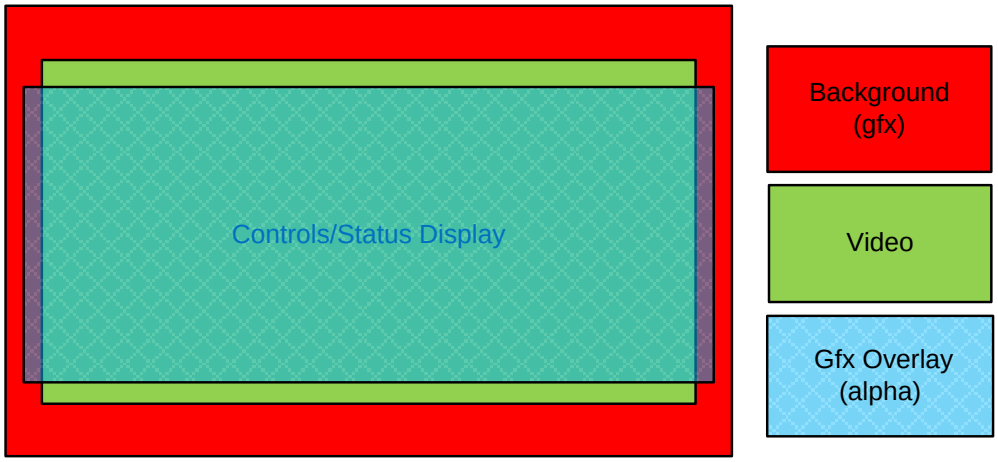
25.3.9.4 Example Use Cases

General Porter-Duff-style blending is not needed in many cases, such as when only one of three windows have per-pixel alpha.

Example 1

In Example 1, window A is a graphics background, window B is a video window, and window C is a graphics overlay on top of video and background. Windows A and B are fully opaque, and window C has per-pixel alpha.

Figure 95: Example 1



Window A weights:

- A and B. Video has precedence so use FIX_WEIGHT with BLEND_WEIGHT=0
- A and C. Graphics overlay has alpha so use DEPENDENT_WEIGHT
- A and B and C. Video has precedence so use FIX_WEIGHT with BLEND_WEIGHT=0

Window B (video) weights:

- A and B: Video has precedence so use FIX_WEIGHT with BLEND_WEIGHT=255
- B and C: Graphics overlay has alpha so use DEPENDENT_WEIGHT
- A and B and C: Graphics overlay has alpha so use DEPENDENT_WEIGHT (since A has BLEND_WEIGHT=0, video has weight 255 – window_C_alpha.)

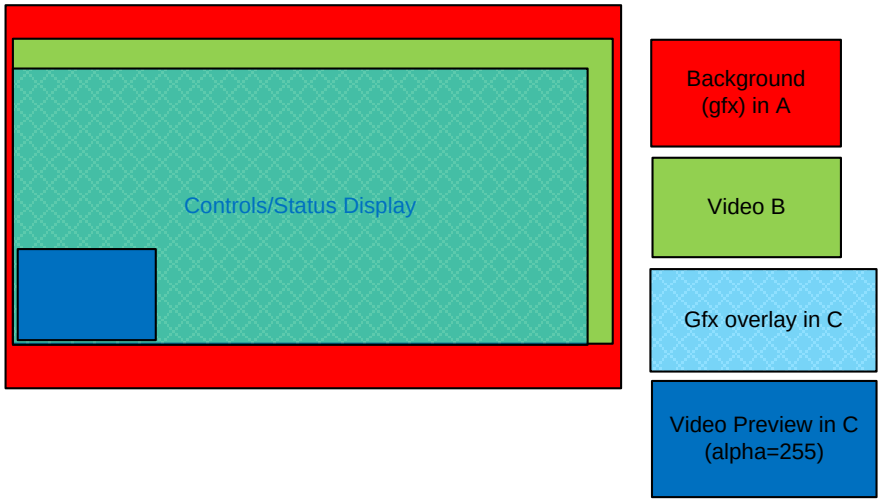
Window C (graphics overlay) weights:

- A and C: There is alpha so use ALPHA_WEIGHT or PREMULT_WEIGHT
- B and C: There is alpha so use ALPHA_WEIGHT or PREMULT_WEIGHT
- A and B and C: There is alpha so use ALPHA_WEIGHT or PREMULT_WEIGHT

Example 2

Example 2 has a similar scenario to Example 1 in which the graphics overlay (with alpha) can be composited by 2D or GPU to have opaque areas (for example, video preview, thumbnails) and transparent areas.

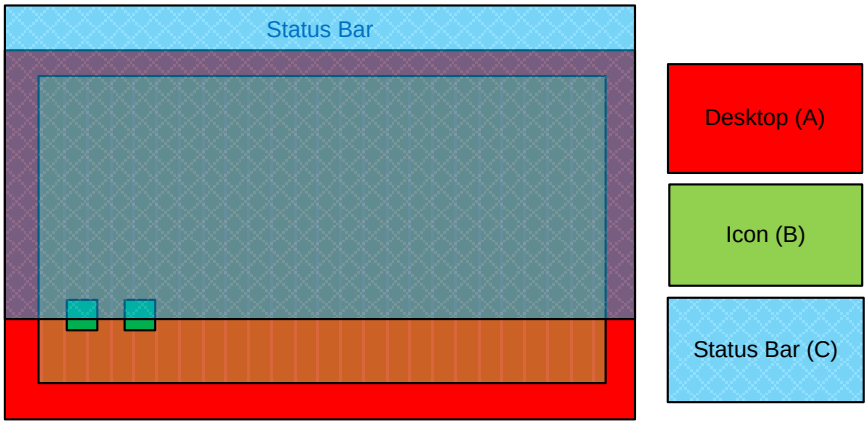
Figure 96: Example 2



Example 3

Example 3 is a phone/tablet use case. The Background (for example, active wallpaper) is opaque graphics. The next layer is icons, perhaps animated. The top layer is the status bar, which can be dragged down to be larger. Both icon and status layers are partially transparent.

Figure 97: Example 3



The two-window overlap cases work fine. The three-window case can use

$$Color_{OUT} = (1 - Alpha_B - Alpha_C) * Color_A + Alpha_B * Color_B + Alpha_C * Color_C$$

Which yields:

- In opaque icon areas, no desktop is shown, as desired. But the resulting color is $Color_B + Alpha_C * Color_C$ rather than the ideal $(1 - Alpha_C) * Color_B + Alpha_C * Color_C$. In other words, the icon color is not dimmed but is diluted by status bar color.
- In transparent icon areas, the desktop will show to the extent allowed by status bar alpha: color is $(1 - Alpha_C) * Color_A + Alpha_C * Color_C$. This is desired.

Window weights for the overlap conditions:

- Window A (desktop) weights:
  - A and B. Icon layer has alpha so use DEPENDENT_WEIGHT
  - A and C. Status bar is transparent so DEPENDENT_WEIGHT
  - A and B and C. DEPENDENT_WEIGHT. Alpha = (255 – icon_alpha – status_bar_alpha).
- Window B (Icon) weights:
  - A and B: There is alpha so use ALPHA_WEIGHT or PREMULT_WEIGHT
  - B and C: Does not happen since the desktop fully overlaps icon
  - A and B and C: ALPHA_WEIGHT
- Window C (status bar) weights.
  - A and C: There is alpha so use ALPHA_WEIGHT or PREMULT_WEIGHT
  - B and C: Does not happen
  - A and B and C: ALPHA_WEIGHT

25.3.9.5 Blending Alternatives

Composition in the 2D unit or with the GPU may be necessary to achieve blending modes not supported by the display controller for three surfaces. Also consider having per-pixel alpha in only one of the windows.

25.3.10 PRISM

This section describes the enhancements to PRISM. PRISM was formerly known as Smart Dimmer, and so the abbreviation SD is commonly used in register names for PRISM.

Features added:

- Programmable statistics window
- Separate crush/adaptation
- Programmable soft clipping
- Increase pixel count limits
- Programmable smooth transitions
- Run histogram on VPULSE2, rather than VSYNC
- LUT/CSC after enhancement

25.3.10.1 Programmable Statistics Window

This feature allows statistics to be gathered over a subset of the active display. It is useful when part of the display is not interesting for PRISM statistics purposes, such as letterbox or border around a video. Without this feature, the border would influence the histogram and therefore the brightness, giving suboptimal results for the video.

The programming interface is a control bit SD_WINDOW_ENABLE in SD_CONTROL register to enable the feature, and two registers containing position and size of the window. When SD_CONTROL == DISABLE (the default), the position/size registers are ignored, and the entire active display region is used, like in Tegra 3.

The coordinate system is relative to display active area – (0,0) is upper left corner of display active. The active region will have upper left corner (H_POSITION, V_POSITION), and lower right corner (H_POSITION + SD_WIN_H_SIZE – 1, V_POSITION + V_SIZE – 1) inclusive. Note this display coordinate system does not take into account surface rotation or device rotation. It is the same coordinate system use by window position (e.g., B_H_POSITION, B_V_POSITION).

The limitation on minimum pixel count of 2^16 must be obeyed.

The programmable window has no direct effect on enhancement – every pixel of display active area is enhanced.

25.3.10.2 Separate Crush/Adaptation

As in Tegra 3 devices, the AGGRESSIVENESS register controlled both the maximum crushed pixel percentage, and the lowest K (maximum gain). This feature separates them and allows software to separately control lowest K (maximum gain). AGRESSIVENESS continues to control the maximum crushed pixel percentage.

The programming interface is a control bit K_LIMIT_ENABLE, and a register SD_K_LIMIT.

As in Tegra 3 devices, the initial K (as determined by accumulating histogram bins until maximum crush pixel percentage is reached). When K_LIMIT_ENABLE=ENABLE, the initial K is clamped to SD_K_LIMIT rather than default “lowest_K”. When K_LIMIT_ENABLE = DISABLE (the default), SD_K_LIMIT register is ignored and the default “lowest_L” are used based on AGRESSIVENESS, as in Tegra 3.

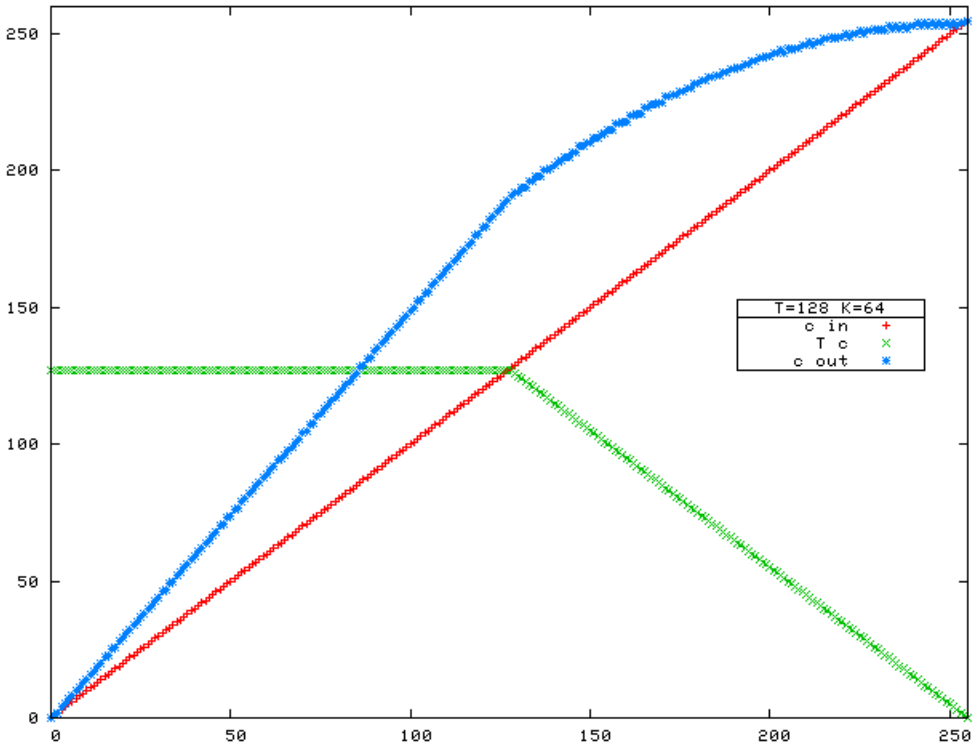
25.3.10.3 Programmable Soft Clipping

In PRISM enhancement, soft clipping is employed to avoid harsh clipping of amplified high-value pixels, and resulting loss of detail in bright areas. The gain applied to high-valued pixels is reduced gradually from 1/K to 1.0, for pixels above a threshold of 128. In Tegra 4 devices, the threshold is fixed at 128, and soft clipping is always enabled. This feature gives more software control.

Since soft clipping is performed after statistics, and it affects the brightness of the image, the statistics and therefore the backlight brightness will be incorrect. This can somewhat compensated for in the backlight transfer-function and/or enhancement value lookup tables.

The Tegra 4 transfer function is shown here, for K=1.5. The T_c plot is the weight applied to fractional portion of K, and goes from 127 to 0, representing 1 to 0.

Figure 98: Default Threshold=128 at K=1.5



With programmable thresholds, other curves are possible. For example, Figure 99 and Figure 100 show more gradual curves using threshold=64 and 0 respectively.

Figure 99: Threshold=64 at K=1.5

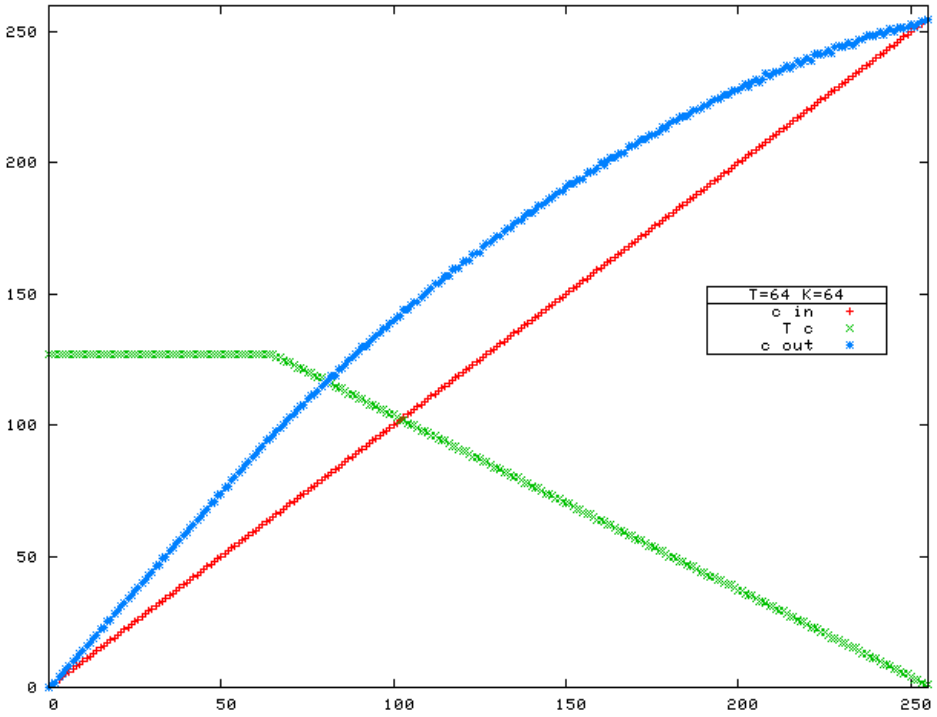
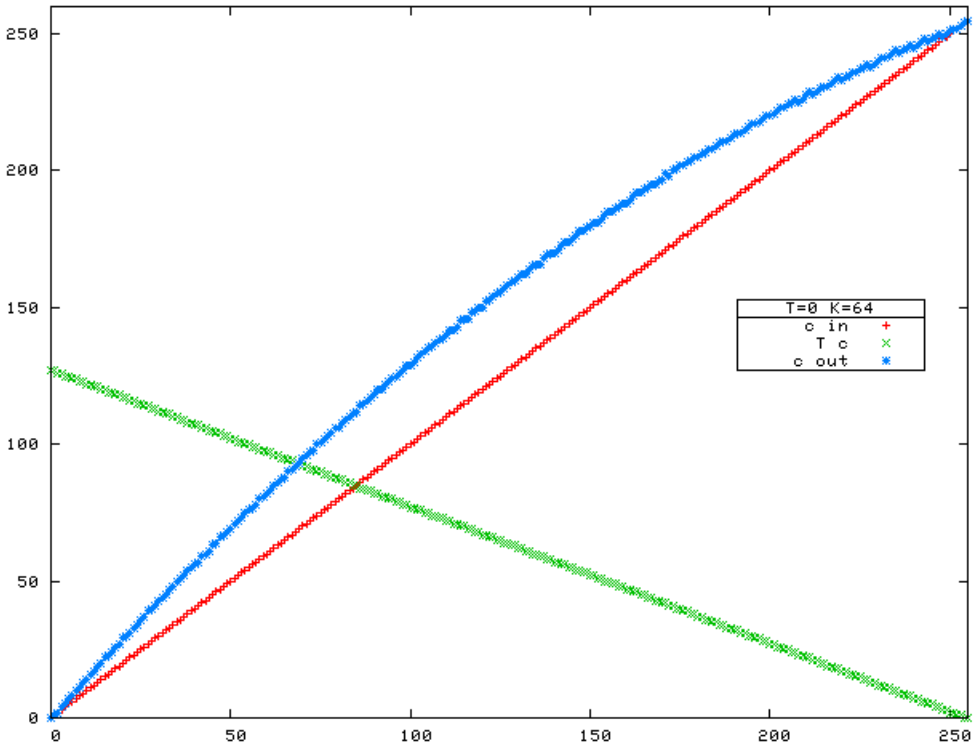


Figure 100: Threshold 0 at K=1.5



Thresholds too high are not usable, because the soft clipping does not kick in early enough to prevent clipping as K grows; note in Figure 101 (threshold 170 at K=1.5) the value of c_out rises about 255. The curve is also not monotonically increasing. Software would need to reduce maximum K to 1.25 or so for such a threshold. Figure 102 shows threshold of 170 with K = 1.25.

Figure 101: Overflowing at Threshold=170 at K=1.5

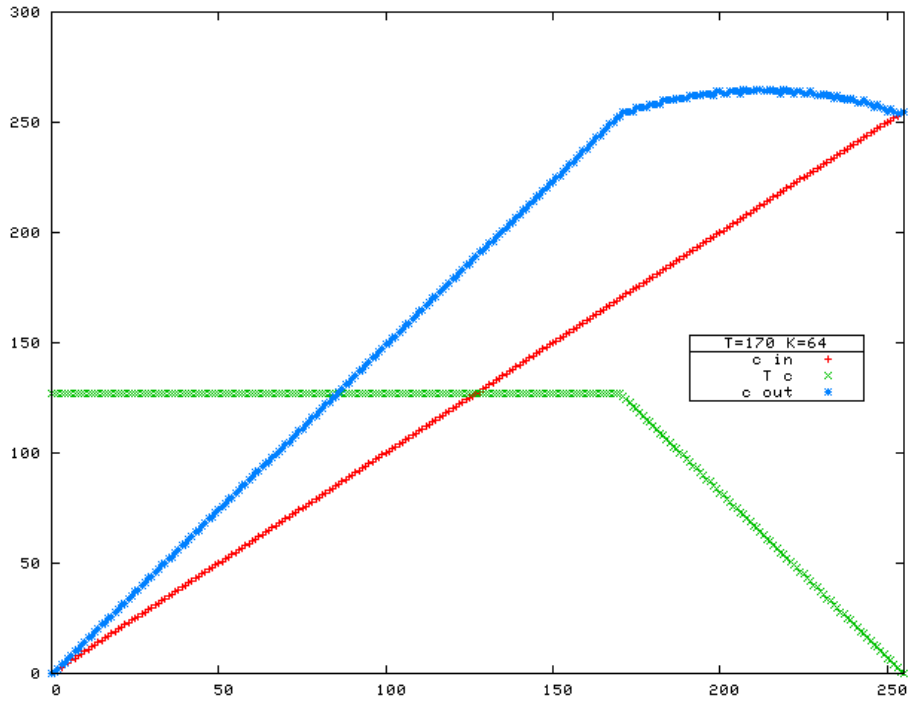
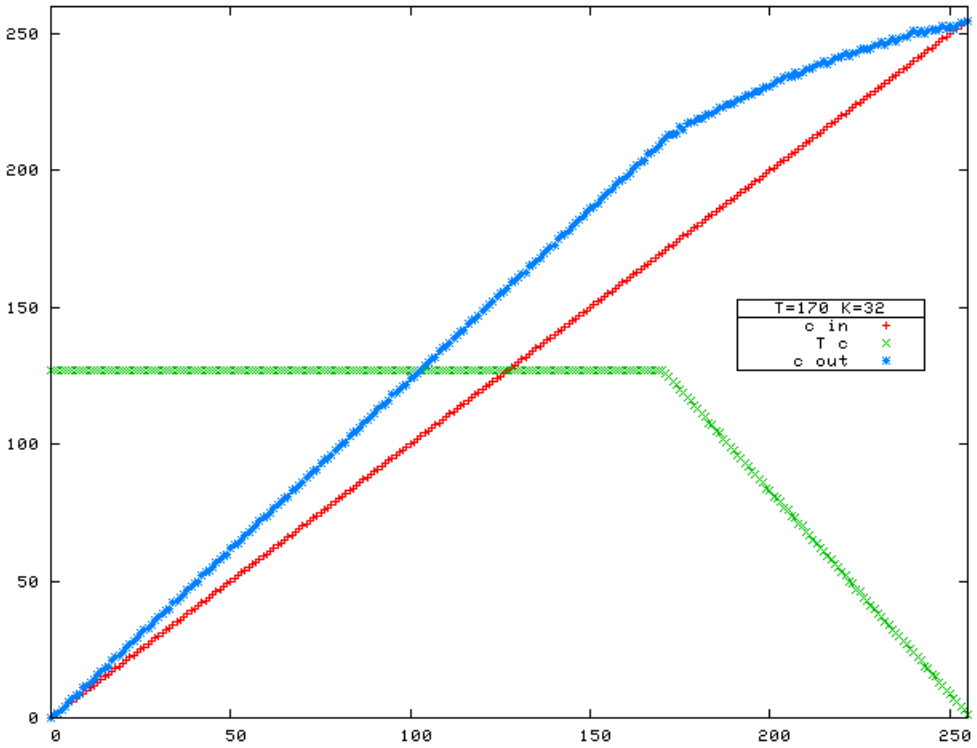


Figure 102: Threshold 170 at K=1.25



The programming interface consists of a control bit SOFT_CLIPPING_ENABLE and one register SD_SOFT_CLIPPING. The latter contains both the threshold and a software-computed reciprocal of inverse threshold. The software-computed reciprocal

avoids the need to perform division in hardware. RECIP is computed as 64*1024 * 1.0 / (256 – threshold), truncated to 16-bit unsigned integer. Internally, the hardware may use fewer than 16 bits as described below.

When SOFT_CLIPPING_ENABLE=DISABLE, then SD_SOFT_CLIPPING is ignored and no clipping is done.

When SOFT_CLIPPING_ENABLE=ENABLE, SD_SOFT_CLIPPING is used. At the default values of SD_SOFT_CLIPPING, it behaves like Tegra 3.

The soft clipping equation performed in hardware, per component, is

```
If SD_CONTROL.SOFT_CLIPPING_ENABLE==ENABLE and c_in >= threshold:
        T_c = 0x7f - (((c_in - threshold) * (recip >> 2)) >> 7);
Else
        T_c = 0x7f
```

Where recip is the programmed SOFT_CLIPPING_RECIP; it is shifted right 2 bits to yield a 14-bit value. T_c is a 7-bit weight. The shift by 7 represents discarding 6 bits from recip plus one to give a 7-bit rather than 8-bit result. As before, T_c is used to scale the fractional portion of K:

```
c_k_limited = (K_c * T_c) >> 7;
```

and then apply the fractional gain to the component:

```
c_mul = (c_in * c_k_limited) >> 7;
```

and finally add the integer portion of gain (which is always 1, i.e., 1 * c_in = c_in):

```
c_out = c_in + c_mul
```

25.3.10.4 Increase Pixel Count Limits

Displays are trending to 2560x1600 and larger panels. Tegra 3 smart dimmer supported a maximum of 2^{21} pixels. As 2560x1600 exceeds that, PRISM now needs to support at least 2^{23}. For simplification, the minimum screen size of 2^{16} is retained. Therefore the pixel count dynamic range increases from 32:1 to 128:1. The total_pixels and histogram accumulator counters must support that range. Also, the histogram normalization must support an expanded range of exponents and shifting (0-7 rather than 0-5).

The SD_HISTOGRAM bins need not expand because they contain normalized pixel percentages, not raw counts.

The programming model does not change. SD_PIXEL_COUNT already accommodates this range.

25.3.10.5 Run Histogram on VPULSE2, rather than VSYNC

In the past, Smart Dimmer histogram processing is done on vsync, which is at the beginning of the frame. This means that a new BRIGHTNESS, based on previous frame's histogram, is not available until the beginning of the next frame. In non-continuous (display self-refresh) mode, there can be a long delay. In Tegra 4 devices, Smart Dimmer can be configured to run on V_PULSE2 instead. This can be programmed to occur at the end of a frame, so that software can get the results in a timely manner, and decide whether to change brightness, retransmit frame with new enhancement value, or neither.

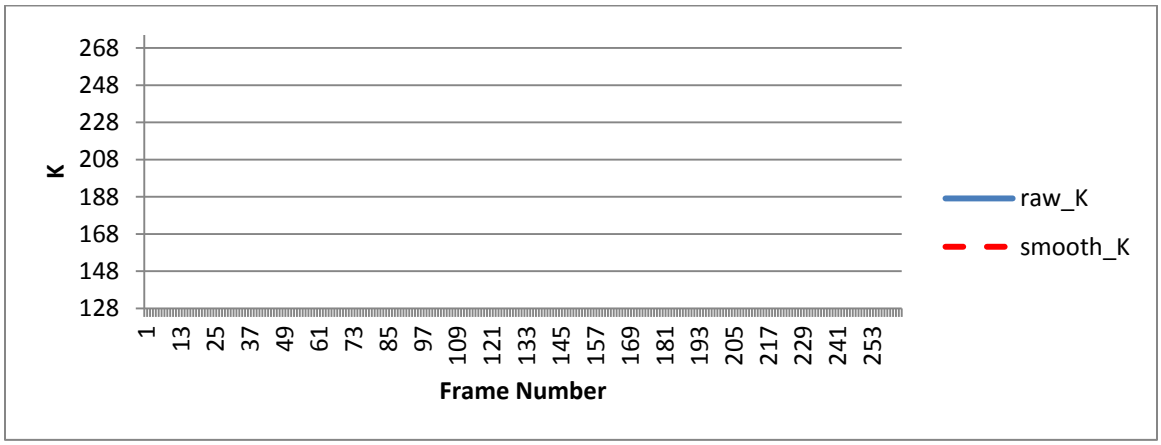
25.3.10.6 Programmable Smooth Transitions

Sudden changes in Smart Dimmer enhancement K and backlight can be noticeable and objectionable. While the soft-clipping compensation algorithm in section on "Programmable Soft Clipping" can help, it may not be enough. To improve user experience, this feature allows changes to be automatically phased in (smoothed) over time.

When enabled, the hardware-generated raw K is the target value. It is not used directly for enhancement or backlight brightness. Rather, a "smooth_K" is used. Each frame, the current smooth_K is incremented or decremented if different than the target raw K. The smooth_K and increment value have four additional fraction bits. The minimum SMOOTH_K_INCR value of 0x1 changes smooth_K by 1/16 per frame. For a maximum raw_K change of 128, the maximum phase-in time is 128*16 = 2048 frames, or about 34 seconds.

Figure 103 illustrates the smooth_k function; SMOOTH_K_INCR = 64 which changes smooth_K by at most 1.0 per frame.

Figure 103: Smooth K



Software can compute SMOOTH_K_INCR as follows. Assume a maximum desired brightness change rate of R L* / second = SMOOTH_K_INCR K/frame. Assume L* is roughly linear with respect to K (L* = K/2). R L* / second = SMOOTH_K_INCR K/frame, and we solve for SMOOTH_K_INCR . R L* / second = R L* / (60 frames) = R K / (2*60 frames) = R/120 K/frame. So SMOOTH_K_INCR = R/120. Because the register is encoded as 8.6 fixed-point, encode as round(R/120 * 64).

25.3.11 Display Color Management Unit (LUT/CSC)

This unit, new to Tegra 4 devices, adds an additional LUT -> CSC -> LUT after the PRISM unit to enable gamma and color gamut change. While this feature is independent from PRISM, and more generally useful, it helps PRISM accommodate non-standard (non-sRGB) panels. It also helps convert from the display's native gamma and gamut (de facto of sRGB) into panels.

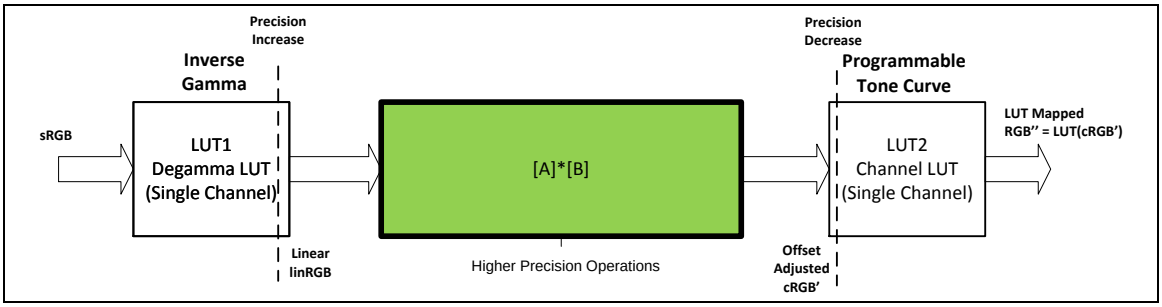
First stage – LUT1 – is an 8-bit to 12-bit RAM-based look-up table (LUT) to convert gamma from sRGB 2.2 to linear (1.0). The output is 8.4 unsigned fixed-point format. A single LUT may be used for all 3 channels.

Second stage – CSC – is 3x3 matrix multiply, to convert color gamut / whitepoint. 12 bits unsigned in, and 12 bits unsigned out. Its nine coefficients are 10-bit (S1.8 signed).

Third stage – LUT2 – is a 12-bit to 8-bit LUT to convert from linear gamma to sRGB or similar gamma (for example, 1 / 2.2). A single LUT may be used for all 3 channels.

> **Note:** The window datapaths have subunits called LUT (CP) and CSC already, but they are for different purposes. The CP is for converting linear framebuffer data into sRGB, or for color index modes, or for applying aesthetic changes like contrast. CSC is for converting YCbCr into RGB. They are for normalizing all surfaces into a common format, nominally sRGB, for blending.

Figure 104: Color Management Unit



25.3.11.1 CMU LUT1

LUT1 maps 8-bit gamma-corrected input to 12-bit linear output. A single LUT may be used for all 3 RGB channels – per-channel control is not required.

```
R′ = LUT1[R]
G′ = LUT1[G]
B′ = LUT1[B]
```

25.3.11.2 CSC Matrix

CSC equation is

$$\begin{bmatrix} KRR & KGR & KBR \\ KRG & KGG & KBG \\ KRB & KGB & KBB \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

25.3.11.3 CMU LUT2

LUT2 maps 12-bit linear input to 8-bit gamma corrected output. A single LUT may be used for all 3 RGB channels – per-channel control is not required. LUT2 is only 960 rows deep, and divided unevenly so that precision is variable; greater precision (more entries) are allocated to darker values. The first 512 entries 0..511, representing range [0.0, 32.0), are darker values, and are mapped with full 9-bit precision (5.4). The top 448 entries are for lower-precision brighter range [32, 256); they are reduced from 8.4 to 8.1 and offset so that values [32,256) map to 512..959 rather than 576..1023

```
R′ = LUT2[zone_select(R)]
G′ = LUT2[zone_select(G)]
B′ = LUT2[zone_select(B)]
```

The zone_select function maps 12 bit component value to 10-bit RAM index, with variable precision:

If C >= THRESHOLD:

C’ = (C >> PRECISION_DIFF) + THRESHOLD – OVERLAP

Else:

C’ = C[9:0]

Where THRESHOLD=2^9 = 512, PRECISION_DIFF = 12 – 9 = 3, and OVERLAP = THRESHOLD >> PRECISION_DIFF = 512 >> 3 = 64.

Software would use an inverse_zone_select to initialize the LUT2:

If C >= THRESHOLD:

C’ = (C - THRESHOLD + OVERLAP) << PRECISION_DIFF

Else:

C’ = C

25.3.11.4 Programming Interface

The CMU LUTs and CSC registers are single-buffered; shadow update and activation have no effect. However, the CMU_ENABLE bit is double-buffered, so it can be enabled/disabled at frame boundaries.

Host interface writes and reads take precedence over the datapath and may result in visible glitches. Software is responsible for ensuring the datapath is idle before writing or reading. An interrupt will be raised upon a conflict. The LUTs are guaranteed to be idle during blanking and when CMU_ENABLE=DISABLE. Software must explicitly enable reading, and reading is only performed for debugging reasons.

Note:	**When reading LUTs:** due to internal RAM latency, after changing read-enable or read address, data is not available until two clocks later. Best way to ensure this is issue two back-to-back writes of same read address, as shown in read sequence.

Software initialization sequence:

```
Insure CMU_ENABLE=DISABLE or display idle
For C = 0 .. 255:
        C’ = ungamma_function(C/255) * 4095
        CMU_LUT1.LUT1_ADDR = C
        CMU_LUT1.LUT1_DATA = C’
        Write CMU_LUT1
For C = 0 .. 959:
        C’ = gamma_function(inverse_zone_select(C)/4095) * 255
        CMU_LUT2.LUT2_ADDR = C
        CMU_LUT2.LUT2_DATA = C’
        Write CMU_LUT2
Write DISP_COLOR_CONTROL.CMU_ENABLE = ENABLE
Use GENERAL_ACT_REQ
```

Where typically functions are: ungamma_function(x) = $x^{2.2}$, and gamma_function(x) = $x^{1/2.2}$.

LUT read sequence:

```
Insure CMU_ENABLE=DISABLE or display idle
For C = 0 .. 255:
        CMU_LUT1_READ.LUT1_READ_EN = 1
        CMU_LUT1_READ.LUT1_READ_ADDR = C
        Write CMU_LUT1_READ
        Write CMU_LUT1_READ # ensure one cycle delay
        Read CMU_LUT1.LUT1_DATA
Write CMU_LUT1_READ.LUT1_READ_EN = 0
```

25.3.11.5 LUT Conflict Interrupt

To help debug conflicts caused by accessing LUTs while datapath is using them, CMU_LUT_CONFLICT interrupt is raised if:

- host write during scanout (pixel read when cmu_enable=1)
- host read during scanout
- host write during host read (i.e; LUT1_READ_EN=1)

25.3.12 Dither

Increases the line buffer from 1280 to 2560 pixels.

The Tegra 4 display uses the following dither methods:

- Error Diffusion
- Dynamic and Static Ordered Dither

25.4 Programming Model

25.4.1 Software Alignment Requirements

For Tegra 4 devices, the pixel data alignment restriction has increased to 64B boundaries. The table below describes what registers are affected.

Parameter	Register name	Pitch (Packed) 1/2/4 Bpp	Pitch (Planar) 1 Bpp per Surface	Tiled (Packed) 1/2/4 Bpp	Tiled (Planar) 1 Bpp per Surface
line_stride	B_LINE_STRIDE	64	64	64	64
h_prescale_size	B_H_PRESCALED_SIZE	Bpp	Even	Bpp	Even
v_prescale_size	B_V_PRESCALED_SIZE	NA	Even for 420	NA	Even for 420
h_offset	B_ADDR_H_OFFSET	Bpp	Even	Bpp	Even
v_offset	B_ADDR_V_OFFSET	NA	Even for 420	NA	Even for 420
Startaddr	B_START_ADDR	64	64	256	256

25.5 Display Controller Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

The display memory map is shown below.

25.5.1 Address Map

Space Name	Offset	Notes
DC_CMD	0x0	Non-shadowed command/sync registers
DC_COM	0x300	Non-shadowed non-window registers
DC_DISP	0x400	Shadowed non-window registers
DC_WINC	0x500	Non-shadowed indirect window registers; All windows; use DISPLAY_WINDOW_HEADER
DC_WIN	0x700	Shadowed indirect window registers; All windows; use DISPLAY_WINDOW_HEADER
DC_WINBUF	0x800	Triple-buffered indirect window registers; All windows; use DISPLAY_WINDOW_HEADER
DC_A_WINC	0xa00	Window A direct alias for DC_WINC
DC_A_WIN	0xb80	Window A direct alias for DC_WIN
DC_A_WINBUF	0xbc0	Window A direct alias for DC_WINBUF
DC_B_WINC	0xc00	Window B direct alias for DC_WINC
DC_B_WIN	0xd80	Window B direct alias for DC_WIN
DC_B_WINBUF	0xdc0	Window B direct alias for DC_WINBUF
DC_C_WINC	0xc00	Window C direct alias for DC_WINC
DC_C_WIN	0xd80	Window C direct alias for DC_WIN
DC_C_WINBUF	0xbc0	Window C direct alias for DC_WINBUF

25.5.2 New Registers for Tegra 4 Devices

The following registers are new or changed for Tegra 4 devices.

Register	Field	Description
B_SURFACE_WEIGHT	B_SURFACE_WEIGHT_OVERRIDE	Override the default weights of the arbiter in display. 0 – disable, 1 - enable

Register	Field	Description
	B_SURFACE_WEIGHT_Y	Surface weight of Y surface
	B_SURFACE_WEIGHT_U	Surface weight of U surface
	B_SURFACE_WEIGHT_V	Surface weight of V surface
B_UFLOW_STATUS	UFLOW_COUNT	Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.
	COUNT_OFLOW	Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
B_UFLOW_CTRL	B_UFLOW_CTRL_DBG_MODE	When the underflow debug mode is enabled. Instead of sending out the last valid pixel, DBG_PIXEL is sent out. 0 – disable, 1 - enable
B_UFLOW_DBG_PIXEL	B_UFLOW_DBG_PIXEL	32-bit debug pixel color.
B_UFLOW_THRESHOLD	B_UFLOW_THRESHOLD	Number of lines to skip when underflow continues to the point where display (XY) is greater than display memfetch request (XY)
B_WIN_OPTIONS	B_SCAN_COLUMN	Load memory columns into display line buffer. 0=disable, 1=enable
B_SCALEFACTOR_ THRESHOLD	B_SF_HWM_THRESHOLD	16-bit high water mark (HWM) value used to compare window line buffer depth against Default – 0xffff
	B_SF_LWM_THRESHOLD	16-bit low water mark (LWM) value used to compare window line buffer depth against Default – 0xffff
B_LATENCY_THRESHOLD	B_RDY4LATENCY_THRESHOLD	Used for DVFS buffering.16-bit HWM value used to compare window line buffer depth against Default – 0xffff
CURSOR_PREFETCH_ LATENCY	SCANLINE_COUNT	5 bits. Number of scan lines prior to cursor first active scan-line when cursor DVFS readiness is disabled. Default: 3
CURSOR_START_ADDR	CURSOR_SIZE	Extend enums to: • 32x32 • 64x64 • 128x128 • 256x256
CURSOR_START_ADDR_ NS	CURSOR_SIZE	Extend enums to: • 32x32 • 64x64 • 128x128 • 256x256
BLEND_CURSOR_ CONTROL	CURSOR_MODE_SELECT	Enums: • Legacy • Normal
SD_K_LIMIT	All	New PRISM feature: Separate Crush/Adaptation; see 25.3.10.2
SD_WINDOW_POSITION	All	New PRISM feature: Programmable Statistics Window; see 25.3.10.1
SD_WINDOW_SIZE	All	As above
SD_SOFT_CLIPPING	All	New PRISM feature: Programmable Soft Clipping; see 25.3.10.3

Register	Field	Description
SD_SMOOTH_K	All	New PRISM feature: Programmable Smooth Transitions; see 25.3.10.6

25.5.2.1 PRISM CMU Registers

For new CMU registers, see the section on Display Color Management Unit (LUT/CSC).

25.5.3 Display Controller Register Definition

Each display supports three windows: Window A, Window B, Window C

The windows may have different features, as described above; however, the register interfaces are the same.

The Display Controller register consists of two template files:

- ardisplay_TEMPLATE.spec: these registers are applicable to all display windows
- ardisplay_b_TEMPLATE.spec: these registers are applicable to windows, A, B, or C. There are three copies of these registers in DISPLAY. Window A, B, and C registers are copies of each other, except for differences in window features. Additionally, the registers can be accessed either indirectly or directly. In indirect mode, the copies of a register in window A, B, and C share the same address; register field DISPLAY_WINDOW_HEADER is used to control whether the subsequent programming goes to window A, B, or C registers, or any combination of them. Before reading these registers, DISPLAY_WINDOW_HEADER must be programmed to enable only one window, so that DISPLAY knows which register copy is being read. In direct mode, the copies in window A, B, and C have separate addresses; DISPLAY_WINDOW_HEADER is ignored.

Color Palette

These appear as three instances of 256 32-bit registers with each register consisting of one RGB pixel so not all 32 bits in the registers are used. One instance is used for window A, another instance is used for window B, and the third instance is used for window C. In reality, each instance is implemented as triple 256-word dual-port register files (2P RF) with one read port and one write port (1R1W) and with each word consisting of 1 red/green/blue component. Read port of the color palettes are used for the corresponding window A, window B, or window C and write port of the color palettes are used for host write.

Note that the host cannot read these color palettes, so it must cache this color palette somewhere else to be able to read them. This is done to reduce area.

25.5.4 Display Shadow Registers

Display registers have three shadow types:

1. Not Shadowed

   Writes to these registers take effect immediately.
2. Double Buffered

   Each register has two copies: ASSEMBLY and ACTIVE. ACTIVE is the working copy.

   These two copies share the same address/offset.

   WRITE_MUX can be programmed to choose which copy the subsequent register writes goes to:
   - When set to ACTIVE, both ACTIVE and ASSEMBLY copies will be written
   - When set to ASSEMBLY, only ASSEMBLY copy will be written

   READ_MUX can be programmed to choose which copy the subsequent register reads comes from.

   If display is in STOP mode, ASSEMBLY copy is latched into ACTIVE copy immediately after

(GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed. In other modes the latching happens on the next frame boundary after (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed.

3. Triple Buffered

   Each register has three copies: ASSEMBLY, ARM, and ACTIVE. ACTIVE is the working copy.

   ASSEMBLY and ACTIVE copies share the same address/offset, the ARM copy is located at the address/offset one bigger than the ASSEMBLY/ACTIVE copy.

   WRITE_MUX can be programmed to choose which copy the subsequent register writes goes to:

   - When set to ACTIVE, all three copies will be written
   - When set to ASSEMBLY, only ASSEMBLY copy will be written

   READ_MUX can be programmed to choose which copy the subsequent register reads comes from.

   - To read back ASSEMBLY or ACTIVE copy, set READ_MUX correctly before register reads.

   These two copies share the same address.

   - To read back ARM copy, do not set READ_MUX, but read back the register/register-field with "_NS" in their names, the offset of which is 1 bigger than its ASSEMBLY/ACTIVE counterpart.

   ASSEMBLY copy is latched into ARM copy immediately after GENERAL/WIN_A/WIN_B/WIN_C)_UPDATE is programmed.

   If display is in STOP mode, ARM copy is latched into ACTIVE copy immediately after (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed. In other modes the latching happens on the next frame boundary after (GENERAL/WIN_A/WIN_B/WIN_C)_ACT_REQ is programmed.

25.5.5 Display Control Modes

The DISPLAY_COMMAND is used to set display control mode (**CONTINUOUS**, **ONE-SHOT**, or **STOP**).

Display switches mode when CONTROL_MODE is programmed to shadow register and then activated.

Display enters a mode when the register activation happens, either on frame boundary when entering **STOP** mode, or immediately when exiting **STOP** mode.

In **STOP** mode, display is in idle. Pixel clock is not running.

In **CONTINUOUS** mode, display keeps refreshing the output frame by frame. This mode is mostly used for the panels without internal frame buffer.

In **ONE-SHOT** mode, also known as non-continuous mode, display waits for a trigger before refreshing each frame. Between those frames, display is in idle. This mode is usually for the panels with internal frame buffer.

In ONE-SHOT mode, there are different types of triggers as follows:

- Input triggers

  Input trigger is to notify DISPLAY that a new memory surface is ready to be displayed.

  - Host Trigger

    Host trigger is requested when NC_HOST_TRIG_ENABLE is programmed. This register field is in the same register as shadow registers UPDATE/ACT_REQ so that the trigger can happen atomically with register updates for the new frame. When host trigger is requested within a frame, the requested frame will start immediately after the end of the current frame.

- Output Triggers

  Output trigger is to notify DISPLAY that a panel is ready to receive data from DISPLAY.

  - Tear Effect Signal Triggers

When MSF/SSF register field is set to ENABLE, DISPLAY holds off a frame requested by input triggers until DISPLAY receives a "ready" from the pins that connect to the tearing effect signals from the panel, then sends a new frame to panel. However, if no input trigger has been requested, DISPLAY does not send a new frame.

In CONTINUOUS mode, the meaning of the triggers is different.

- Input Triggers

  Since in CONTINUOUS mode trigger happen automatically and repeatedly, trigger here only affects the buffer address change on a window.

  - Host Trigger

    Host can change the buffer address and then write WIN_A/B/C_ACT_REQ to request the buffer address change. Note that for syncpt logic, it cannot tell if a WIN_A/B/C register activation indicates the change of address, so it always behaves like a change of address happens on any register activation requested by WIN_A/B/C_ACT_REQ. As the result, RD_DONE/OP_DONE is returned.

- Output triggers

  These triggers are not applicable in CONTINUOUS mode. MSF/SSF_ENABLE should never be set in CONTINUOUS mode.

25.5.6 Sync Points

DISPLAY has 4 syncpt clients:

- GENERAL

  Conditions

  - 0: IMMEDIATE return INDX immediately
  - 1: OP_DONE not meaningful, same as IMMEDIATE
  - 2: RD_DONE not meaningful, same as IMMEDIATE
  - 3: REG_WR_SAFE returns INDX whenever it is safe to program GENERAL shadow registers (when all previous GENERAL activation has happened)
  - 4: HSPI returns INDX whenever it is safe to send HSPI data (when all HSPI_*** registers are safe to be programmed)
  - 5: FRAME_DONE returns INDX on the next frame end
  - 6: VPULSE3 returns INDX on the next vpulse3 leading edge (for timing sensitive operations if needed)
  - 7: FRAME_START returns INDX on the next frame start (currently for hardware testing, but can be used if needed)

- WIN_A

  Conditions

  - 0: IMMEDIATE return INDX immediately
  - 1: OP_DONE same as RD_DONE
  - 2: RD_DONE returns INDX when all WIN_A reads for frames activated before the INCR_SYNCPT are complete. In ONE-SHOT mode with TVO disabled, this happens after the last row of window A display. In CONTINUOUS mode, or in ONE-SHOT mode with TVO enabled, this happens on the last row of window A or on frame end (depending if the INCR_SYNCPT is issued before or after the window A last row).
  - 3: REG_WR_SAFE returns INDX whenever it is safe to program WIN_A shadow registers (when all previous WIN_A activation has happened)

- WIN_B

  Similar to WIN_A

- WIN_C

  Similar to WIN_A
- DISPLAY Continuous syncpt

  VSYNC: When enabled, return syncpt whenever a vblank start happens.

25.5.7 Raise Actions (Legacy)

The Raise action is used to check the state of the display's command execution. The Raise action can be enabled by itself or with other actions.

The raise value is returned to the host controller if the current and all previous commands have completed execution. If the command is executed with the delayed mode, the raise is not returned till that command is executed at the end of the frame.

If there is no pending command when the Raise is issued, the raise value is returned immediately. The raise value is a 4-bit number included in the command.

Context switch acknowledge, register reads, and raises are returned to the host through a read FIFO. However, because there may be more than one raise, read, or acknowledge available for writing to the FIFO in a certain cycle, there must be a priority encoder. Here is the priority:

1. Context switch
2. Register read
3. SIGNAL_RAISE
4. SIGNAL_RAISE1
5. SIGNAL_RAISE2
6. SIGNAL_RAISE3
7. DISP_COMMAND_RAISE
8. HSPI_RAISE
9. Refcount

These packets are for verification; they have no relevance to software. The RAISE packet should include the intended return channel. This channel should be passed back to the host so the RAISE will be reflected in the correct channel.

25.6 Display CMD Registers

25.6.1 DC_CMD_GENERAL_INCR_SYNCPT_0

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description

Bit	Reset	Description
15:8	0x0	GENERAL_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = HSPI 5 = FRAME_DONE 6 = VPULSE3 7 = FRAME_START 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	GENERAL_INDX: syncpt index value

25.6.2 DC_CMD_GENERAL_INCR_SYNCPT_CNTRL_0

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	GENERAL_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	GENERAL_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal state of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

25.6.3 DC_CMD_GENERAL_INCR_SYNCPT_ERROR_0

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	GENERAL_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

25.6.4 DC_CMD_WIN_A_INCR_SYNCPT_0

Offset: 0x8 | Byte Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description

Bit	Reset	Description
15:8	0x0	WIN_A_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	WIN_A_INDX: syncpt index value

25.6.5 DC_CMD_WIN_A_INCR_SYNCPT_CNTRL_0

Offset: 0x9 | Byte Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	WIN_A_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	WIN_A_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

25.6.6 DC_CMD_WIN_A_INCR_SYNCPT_ERROR_0

Offset: 0xa | Byte Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	WIN_A_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

25.6.7 DC_CMD_WIN_B_INCR_SYNCPT_0

Offset: 0x10 | Byte Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description

Bit	Reset	Description
15:8	0x0	WIN_B_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	WIN_B_INDX: syncpt index value

25.6.8 DC_CMD_WIN_B_INCR_SYNCPT_CNTRL_0

Offset: 0x11 | Byte Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	WIN_B_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	WIN_B_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

25.6.9 DC_CMD_WIN_B_INCR_SYNCPT_ERROR_0

Offset: 0x12 | Byte Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	WIN_B_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

25.6.10 DC_CMD_WIN_C_INCR_SYNCPT_0

Offset: 0x18 | Byte Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description

Bit	Reset	Description
15:8	0x0	WIN_C_COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	WIN_C_INDX: syncpt index value

25.6.11 DC_CMD_WIN_C_INCR_SYNCPT_CNTRL_0

Offset: 0x19 | Byte Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	WIN_C_INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	WIN_C_INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all Host1x clients affected, then clear all SOFT_RESETs.

25.6.12 DC_CMD_WIN_C_INCR_SYNCPT_ERROR_0

Offset: 0x1a | Byte Offset: 0x68 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	WIN_C_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

25.6.13 DC_CMD_CONT_SYNCPT_VSYNC_0

Offset: 0x28 | Byte Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
8	0x0	VSYNC_EN: On host read bus every time VSYNC (V-blank leading edge) happens and VSYNC_EN is set 0 = DISABLE 1 = ENABLE
7:0	X	VSYNC_INDX: Return INDX (set HOST_CLRD packet TYPE field to SYNCPT)

25.6.14 DC_CMD_CTXSW_0

Context switch registers for class and channel

Should be common to all modules. Includes the current channel/class (writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Context switch works like this:

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts.

Another way to avoid context switch interrupts is to set the AUTO_ACK bit.

This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x30 | Byte Offset: 0xc0 | Read/Write: R/W | Reset: 0xf000f800 (0bxxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

25.6.15 DC_CMD_DISPLAY_COMMAND_OPTION0_0

Display Controller Option 0. This register is not effective until DISPLAY_COMMAND is written.

Class: Display Command

Offset: 0x31 | Byte Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx000xxxxxxx00000000)

Bit	Reset	Description
18	0x0	WINDOW_C_NC_DISPLAY: Window C Non-Continuous Display. This is effective only in Non-Continuous Display mode when window B buffer switching is not controlled by host. If this bit is enabled, a frame is sent whenever Window B buffer is switched. 0 = DISABLE 1 = ENABLE
17	0x0	WINDOW_B_NC_DISPLAY: Window B Non-Continuous Display. This is effective only in Non-Continuous Display mode when window B buffer switching is not controlled by host. If this bit is enabled, a frame is sent whenever Window B buffer is switched. 0 = DISABLE 1 = ENABLE
16	0x0	WINDOW_A_NC_DISPLAY: Window A Non-Continuous Display. This is effective only in Non-Continuous Display mode when window A buffer switching is not controlled by host. If this bit is enabled, a frame is sent whenever Window A buffer is switched. 0 = DISABLE 1 = ENABLE
7:6	0x0	SSF_SOURCE: Source pin for the SSF input. Controls which pin will be used as the source for the trigger input when MSF mode is enabled. Note that although the same pins are available for both MSF and SSF, the order in the enum and hence the values differ between the pins. This is to maintain backwards compatibility with previous chips, which had a fixed mapping. The init value and the first value in the enum reflect this historical mapping. 0= LCD_DC pin (legacy default) 1= LCD_SPI pin 2= LCD_SDI pin 3= RESERVED for future use. 0 = SSF_LDC 1 = SSF_LSPI 2 = SSF_LSDI
5	0x0	SSF_ENABLE: Sub-Display Stop Frame (SSF) input. This is effective only in Non-Continuous Display mode. When enabled, SSF signal can be input through LDC pin. When SSF is enabled a trigger to send a frame in Non-Continuous Display mode will be delayed until SSF is active. 0 = DISABLE 1 = ENABLE
4	0x0	SSF_POLARITY: Sub-Display Stop Frame (SSF) Polarity 0= Active high 1= Active low
3:2	0x0	MSF_SOURCE: Source pin for the MSF input. Controls which pin will be used as the source for the trigger input when MSF mode is enabled. Note that although the same pins are available for both MSF and SSF, the order in the enum and hence the values differ between the pins. This is to maintain backwards compatibility with previous chips, which had a fixed mapping. The init value and the first value in the enum reflects this historical mapping. 0= LCD_SPI pin (legacy default) 1= LCD_DC pin 2= LCD_SDI pin 3= RESERVED for future use. 0 = MSF_LSPI 1 = MSF_LDC 2 = MSF_LSDI
1	0x0	MSF_ENABLE: Main-Display Stop Frame (MSF) input This is effective only in Non-Continuous Display mode. When enabled, the MSF signal can be input through the LSPI pin. When MSF is enabled, a trigger to send a frame in Non-Continuous Display mode will be delayed until MSF is active. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	MSF_POLARITY: Main-Display Stop Frame (MSF) Polarity 0= Active high 1= Active low

25.6.16 DC_CMD_DISPLAY_COMMAND_0

Display Command

Offset: 0x32 | Byte Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00xxxx0)

Bit	Reset	Description
30:27	X	DISP_COMMAND_RAISE_CHANNEL_ID: Display Command Channel ID
26:22	X	DISP_COMMAND_RAISE_VECTOR: Display Command Raise Vector. This raise vector is at the next line or frame boundary, depending on GENERAL_ACT_CNTR_SEL
6:5	0x0	DISPLAY_CTRL_MODE: Display Controller Mode 0= Stop Display, this can be used to stop sending frame at the next frame boundary. This is automatically generated in Non-Continuous Display after sending one frame. If this is issued when display controller is already stopped then there is no frame sent. Raise vector (if raise is enabled) is also returned immediately. This command can also be used in non-continuous display mode to stop accepting non-host trigger conditions from other clients. 1= Continuous Display, the display controller will continuously send frame. Continuous display mode can be stopped by switching to Non-Continuous Display or by issuing Stop Display. 2= Non-Continuous Display, the display controller is forced to send one frame of each active display and then wait for the next time this command is issued or for other (non-host) trigger conditions to send frame. The sending of frames may be delayed by MSF or SSF input signals from the display device. If a Stop Display is issued while in non-continuous display mode then non-host trigger conditions will no longer be accepted until the next time Non-Continuous Display is issued 0 = STOP 1 = C_DISPLAY 2 = NC_DISPLAY
0	0x0	DISP_COMMAND_RAISE: Display Command Raise. Raise vector will be returned at the end of command completion 0 = DISABLE 1 = ENABLE

25.6.17 DC_CMD_SIGNAL_RAISE_0

When written, the next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE_TYPE option so that multiple raises can be returned without software intervention. SIGNAL_RAISE1, SIGNAL_RAISE2, or SIGNAL_RAISE3 can be used if more than one source signal is needed.

Offset: 0x33 | Byte Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
19:16	X	SIGNAL_RAISE_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE_TYPE: 0= One-shot, single raise returned. 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE_SELECT=NONE or SIGNAL_RAISE_TYPE=ONESHOT 0 = ONESHOT 1 = CONT

Bit	Reset	Description
10:8	X	SIGNAL_RAISE_SELECT: which signal to raise on 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE_VECTOR: bit number to raise

25.6.18 DC_CMD_DISPLAY_POWER_CONTROL_0

Display Power Control

Offset: 0x36 | Byte Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx00xxxxx0x0xxxxxxx0x0x0x0x0)

Bit	Reset	Description
25	0x0	HSPI_ENABLE: Host SPI write cycle enable. SPI_ENABLE must be enabled also for this bit to be effective. 0 = DISABLE 1 = ENABLE
24	0x0	SPI_ENABLE: SPI interface enable. This enables clock to SPI interface logic for Host SPI, IS SPI, and LCD SPI. 0 = DISABLE 1 = ENABLE
18	0x0	PM1_ENABLE: PM1 signal enable 0 = DISABLE 1 = ENABLE
16	0x0	PM0_ENABLE: PM0 signal enable 0 = DISABLE 1 = ENABLE
8	0x0	PW4_ENABLE: PW4 signal enable. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE
6	0x0	PW3_ENABLE: PW3 signal enable. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE
4	0x0	PW2_ENABLE: PW2 signal enable. This signal controls pixel data processing. It should be enabled during V blank time. This signal also controls the time when pin polarity takes effect at the pad. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	PW1_ENABLE: PW1 signal enable. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE
0	0x0	PW0_ENABLE: PW0 signal enable. This signal controls the display H and V counters. It must be enabled first and disabled last during display power sequencing. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE

25.6.19 DC_CMD_INT_STATUS_0

This reflects status of all pending interrupts which is valid as long as the interrupt is not cleared even if the interrupt is masked. A pending interrupt can be cleared by writing a '1' to this the corresponding interrupt status bit in this register.

Display Interrupt and Status

Offset: 0x37 | Byte Offset: 0xdc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23	X	HC_UF_INT: Cursor Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
22	X	CMU_LUT_CONFLICT_INT: CMU LUT read/write conflict; write 1 to clear
16	X	WIN_C_OF_INT: Window C Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
15	X	WIN_B_OF_INT: Window B Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
14	X	WIN_A_OF_INT: Window A Overflow Interrupt Status 0= interrupt not pending 1= interrupt pending
13	X	SSF_INT: Sub-Display Stop Frame Interrupt Status 0= interrupt not pending 1= interrupt pending
12	X	MSF_INT: Main-Display Stop Frame Interrupt Status 0= interrupt not pending 1= interrupt pending
10	X	WIN_C_UF_INT: Window C Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
9	X	WIN_B_UF_INT: Window B Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending

Bit	Reset	Description
8	X	WIN_A_UF_INT: Window A Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
7	X	SPI_BUSY_INT: SPI Busy Interrupt Status 0= interrupt not pending 1= interrupt pending
5	X	V_PULSE2_INT: Vertical Pulse 2 Interrupt 0= interrupt not pending 1= interrupt pending
4	X	V_PULSE3_INT: Vertical Pulse 3 Interrupt 0= interrupt not pending 1= interrupt pending
3	X	H_BLANK_INT: Horizontal Blank Interrupt 0= interrupt not pending 1= interrupt pending
2	X	V_BLANK_INT: Vertical Blank Interrupt 0= interrupt not pending 1= interrupt pending
1	X	FRAME_END_INT: Frame End Interrupt 0= interrupt not pending 1= interrupt pending
0	X	CTXSW_INT: Context Switch Interrupt Status (this is cleared on write) 0= interrupt not pending 1= interrupt pending

25.6.20 DC_CMD_INT_MASK_0

Setting bits in this register masks the corresponding interrupt but neither clears a pending interrupt nor prevents a pending interrupt from being generated. Masking an interrupt also does not clear a pending interrupt status and does not prevent a pending interrupt status from being generated.

Interrupt Mask

Offset: 0x38 | Byte Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00xxxxx00000x0000x000000)

Bit	Reset	Description
23	0x0	HC_UF_INT_MASK: Cursor Underflow Interrupt Mask. 0 = MASKED 1 = NOTMASKED
22	0x0	CMU_LUT_CONFLICT_INT_MASK: CMU LUT read/write conflict. 0 = MASKED 1 = NOTMASKED
16	0x0	WIN_C_OF_INT_MASK: Window C Overflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
15	0x0	WIN_B_OF_INT_MASK: Window B Overflow Interrupt Mask 0 = MASKED 1 = NOTMASKED

Bit	Reset	Description
14	0x0	WIN_A_OF_INT_MASK: Window A Overflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
13	0x0	SSF_INT_MASK: Sub-Display Stop Frame Interrupt Mask 0 = MASKED 1 = NOTMASKED
12	0x0	MSF_INT_MASK: Main-Display Stop Frame Interrupt Mask 0 = MASKED 1 = NOTMASKED
10	0x0	WIN_C_UF_INT_MASK: Window C Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
9	0x0	WIN_B_UF_INT_MASK: Window B Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
8	0x0	WIN_A_UF_INT_MASK: Window A Underflow Interrupt Mask 0 = MASKED 1 = NOTMASKED
7	0x0	SPI_BUSY_INT_MASK: SPI Busy Interrupt Mask 0 = MASKED 1 = NOTMASKED
5	0x0	V_PULSE2_INT_MASK: Vertical Pulse 2 Interrupt Mask 0 = MASKED 1 = NOTMASKED
4	0x0	V_PULSE3_INT_MASK: Vertical Pulse 3 Interrupt Mask 0 = MASKED 1 = NOTMASKED
3	0x0	H_BLANK_INT_MASK: Horizontal Blank Interrupt Mask 0 = MASKED 1 = NOTMASKED
2	0x0	V_BLANK_INT_MASK: Vertical Blank Interrupt Mask 0 = MASKED 1 = NOTMASKED
1	0x0	FRAME_END_INT_MASK: Frame End Interrupt Mask 0 = MASKED 1 = NOTMASKED
0	0x0	CTXSW_INT_MASK: Context Switch Interrupt Mask 0 = MASKED 1 = NOTMASKED

25.6.21 DC_CMD_INT_ENABLE_0

Setting bits in this register enable the corresponding interrupt event to generate a pending interrupt. Interrupt output signal will be activated only if the corresponding interrupt is not masked.

Disabling an interrupt will not clear a corresponding pending interrupt - it only prevents a new interrupt event to generate a pending interrupt.

Interrupt Enable

Offset: 0x39 | Byte Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxx00xxxx00000x0000x000001)

Bit	Reset	Description
23	0x0	HC_UF_INT_ENABLE: Cursor Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
22	0x0	CMU_LUT_CONFLICT_INT_ENABLE: CMU LUT read/write conflict 0 = DISABLE 1 = ENABLE
16	0x0	WIN_C_OF_INT_ENABLE: Window C Overflow Interrupt Enable 0 = DISABLE 1 = ENABLE
15	0x0	WIN_B_OF_INT_ENABLE: Window B Overflow Interrupt Enable 0 = DISABLE 1 = ENABLE
14	0x0	WIN_A_OF_INT_ENABLE: Window A Overflow Interrupt Enable 0 = DISABLE 1 = ENABLE
13	0x0	SSF_INT_ENABLE: Sub-Display Stop Frame Interrupt Enable 0 = DISABLE 1 = ENABLE
12	0x0	MSF_INT_ENABLE: Main-Display Stop Frame Interrupt Enable 0 = DISABLE 1 = ENABLE
10	0x0	WIN_C_UF_INT_ENABLE: Window C Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
9	0x0	WIN_B_UF_INT_ENABLE: Window B Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
8	0x0	WIN_A_UF_INT_ENABLE: Window A Underflow Interrupt Enable 0 = DISABLE 1 = ENABLE
7	0x0	SPI_BUSY_INT_ENABLE: SPI Busy Interrupt Enable 0 = DISABLE 1 = ENABLE
5	0x0	V_PULSE2_INT_ENABLE: Vertical Pulse 2 Interrupt Enable 0 = DISABLE 1 = ENABLE
4	0x0	V_PULSE3_INT_ENABLE: Vertical Pulse 3 Interrupt Enable 0 = DISABLE 1 = ENABLE
3	0x0	H_BLANK_INT_ENABLE: Horizontal Blank Interrupt Enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	V_BLANK_INT_ENABLE: Vertical Blank Interrupt Enable 0 = DISABLE 1 = ENABLE
1	0x0	FRAME_END_INT_ENABLE: Frame End Interrupt Enable 0 = DISABLE 1 = ENABLE
0	0x1	CTXSW_INT_ENABLE: Context Switch Interrupt Enable 0 = DISABLE 1 = ENABLE

25.6.22 DC_CMD_INT_TYPE_0

Two interrupt types are available:

- Edge interrupt - transition on input signal/event generates pending interrupt
- Level interrupt - active level on input signal/event generates pending interrupt

Interrupt Type

Offset: 0x3a | Byte Offset: 0xe8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00xxxxx00000x0000x00000x)

Bit	Reset	Description
23	0x0	HC_UF_INT_TYPE: Cursor Underflow Interrupt Type 0 = EDGE 1 = LEVEL
22	0x0	CMU_LUT_CONFLICT_INT_TYPE: CMU LUT read/write conflict. Must be EDGE. 0 = EDGE 1 = LEVEL
16	0x0	WIN_C_OF_INT_TYPE: Window C Overflow Interrupt Type 0 = EDGE 1 = LEVEL
15	0x0	WIN_B_OF_INT_TYPE: Window B Overflow Interrupt Type 0 = EDGE 1 = LEVEL
14	0x0	WIN_A_OF_INT_TYPE: Window A Overflow Interrupt Type 0 = EDGE 1 = LEVEL
13	0x0	SSF_INT_TYPE: Sub-Display Stop Frame Interrupt Type 0 = EDGE 1 = LEVEL
12	0x0	MSF_INT_TYPE: Main-Display Stop Frame Interrupt Type 0 = EDGE 1 = LEVEL
10	0x0	WIN_C_UF_INT_TYPE: Window C Underflow Interrupt Type 0 = EDGE 1 = LEVEL

Bit	Reset	Description
9	0x0	WIN_B_UF_INT_TYPE: Window B Underflow Interrupt Type 0 = EDGE 1 = LEVEL
8	0x0	WIN_A_UF_INT_TYPE: Window A Underflow Interrupt Type 0 = EDGE 1 = LEVEL
7	0x0	SPI_BUSY_INT_TYPE: SPI Busy Interrupt Type 0 = EDGE 1 = LEVEL
5	0x0	V_PULSE2_INT_TYPE: Vertical Pulse 2 Interrupt Type 0 = EDGE 1 = LEVEL
4	0x0	V_PULSE3_INT_TYPE: Vertical Pulse 3 Interrupt Type 0 = EDGE 1 = LEVEL
3	0x0	H_BLANK_INT_TYPE: Horizontal Blank Interrupt Type 0 = EDGE 1 = LEVEL
2	0x0	V_BLANK_INT_TYPE: Vertical Blank Interrupt Type 0 = EDGE 1 = LEVEL
1	0x0	FRAME_END_INT_TYPE: Frame End Interrupt Type 0 = EDGE 1 = LEVEL

25.6.23 DC_CMD_INT_POLARITY_0

For edge interrupts, these bits specify whether a pending interrupt is generated on the falling edge or on the rising edge of the corresponding input signal/event. For level interrupts, these bits specify whether a pending interrupt is generated on the low level or on the high level of the corresponding input signal/event. 0 rw CTXSW_INT_POLARITY init=0 // Context Switch Interrupt Polarity enum (LOW, HIGH) // 0= falling edge or low level interrupt // 1= rising edge or high level interrupt.

Interrupt Polarity

Offset: 0x3b | Byte Offset: 0xec | Read/Write: R/W | Reset: 0x00400000 (0bxxxxxxxx01xxxxx00000x0000x00000x)

Bit	Reset	Description
23	0x0	HC_UF_INT_POLARITY: Cursor Underflow Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
22	HIGH	CMU_LUT_CONFLICT_INT_POLARITY: CMU LUT read/write conflict. Must be HIGH. 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH

Bit	Reset	Description
16	0x0	WIN_C_OF_INT_POLARITY: Window C Overflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
15	0x0	WIN_B_OF_INT_POLARITY: Window B Overflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
14	0x0	WIN_A_OF_INT_POLARITY: Window A Overflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
13	0x0	SSF_INT_POLARITY: Sub-Display Stop Frame. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
12	0x0	MSF_INT_POLARITY: Main-Display Stop Frame. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
10	0x0	WIN_C_UF_INT_POLARITY: Window C Underflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
9	0x0	WIN_B_UF_INT_POLARITY: Window B Underflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
8	0x0	WIN_A_UF_INT_POLARITY: Window A Underflow. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
7	0x0	SPI_BUSY_INT_POLARITY: SPI Busy. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH

Bit	Reset	Description
5	0x0	V_PULSE2_INT_POLARITY: V Pulse 2 Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
4	0x0	V_PULSE3_INT_POLARITY: V Pulse 3. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
3	0x0	H_BLANK_INT_POLARITY: H Blank. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
2	0x0	V_BLANK_INT_POLARITY: V Blank. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
1	0x0	FRAME_END_INT_POLARITY: Frame End. Interrupt Polarity 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH

25.6.24 DC_CMD_SIGNAL_RAISE1_0

When written, next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE1_TYPE option so that multiple raises can be returned without software intervention.

Offset: 0x3c | Byte Offset: 0xf0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
19:16	X	SIGNAL_RAISE1_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE1_TYPE: 0= One-shot, single raise returned. 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE1_SELECT=NONE or SIGNAL_RAISE1_TYPE=ONESHOT 0 = ONESHOT 1 = CONT

Bit	Reset	Description
10:8	X	SIGNAL_RAISE1_SELECT: which signal to raise on. 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE1_VECTOR: bit number to raise

25.6.25 DC_CMD_SIGNAL_RAISE2_0

When written, next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE2_TYPE option so that multiple raises can be returned without software intervention.

Offset: 0x3d | Byte Offset: 0xf4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
19:16	X	SIGNAL_RAISE2_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE2_TYPE: 0= One-shot, single raise returned 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE2_SELECT=NONE or SIGNAL_RAISE2_TYPE=ONESHOT 0 = ONESHOT 1 = CONT
10:8	X	SIGNAL_RAISE2_SELECT: which signal to raise on 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE2_VECTOR: bit number to raise

25.6.26 DC_CMD_SIGNAL_RAISE3_0

When written, next occurrence of the selected SIGNAL will cause a RAISE to be sent to the host. Software must not write this register if a previous request (from previous write) is still outstanding. Added SIGNAL_RAISE3_TYPE option so that multiple raises can be returned without software intervention.

Offset: 0x3e | Byte Offset: 0xf8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0xxxxxxxxxxx)

Bit	Reset	Description
19:16	X	SIGNAL_RAISE3_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE3_TYPE: 0= One-shot, single raise returned. 1= Continuous, raise is returned persistently whenever raise event is true until this register is reprogrammed such that SIGNAL_RAISE3_SELECT=NONE or SIGNAL_RAISE3_TYPE=ONESHOT 0 = ONESHOT 1 = CONT
10:8	X	SIGNAL_RAISE3_SELECT: which signal to raise on 0= none, no raise sent back 1= Frame End signal 2= V Blank signal 3= V Pulse 3 signal 4= Rising edge of V Blank signal 5= Falling edge of V Blank signal 6= Rising edge of V Pulse 3 signal 7= Falling edge of V Pulse 3 signal 0 = NONE 1 = FRAME_END 2 = VBLANK 3 = VPULSE3 4 = VBLANK_START 5 = VBLANK_END 6 = VPULSE3_START 7 = VPULSE3_END
4:0	X	SIGNAL_RAISE3_VECTOR: bit number to raise

25.6.27 DC_CMD_STATE_ACCESS_0

Double/triple buffers read and write access control

Offset: 0x40 | Byte Offset: 0x100 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0x0)

Bit	Reset	Description
2	0x0	WRITE_MUX: Write access control 0= write assembly state 1= write active state When set to ACTIVE, register writes also propagate to assembly set for double buffered registers, to both assembly and arm set for triple buffered registers. 0 = ASSEMBLY 1 = ACTIVE

Bit	Reset	Description
0	0x0	READ_MUX: Read access control 0= read assembly state 1= read active state ARM state register read is not controlled by this mux, but by reading the registers with "_NS" suffix 0 = ASSEMBLY 1 = ACTIVE

25.6.28 DC_CMD_STATE_CONTROL_0

State Control for activating/arming new register state

Restrictions: ACT_REQ cannot be programmed at the same time the corresponding "UPDATE" is programmed.

If so desired, it should be split into two consecutive writes to this register.

Offset: 0x41 | Byte Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx0xxxxxxxxxxx0000xxxx0000)

Bit	Reset	Description
24	0x0	NC_HOST_TRIG_ENABLE: Host trigger enable. Effective only in Non-continuous mode. The exception is that when TVO is enabled, this trigger is ignored so as not to corrupt TV output. Note that when this field is enabled, GENERAL_ACT_REQ must be enabled at the same time. 0= disable: no frame is triggered 0 = DISABLE 1 = ENABLE
11	0x0	WIN_C_UPDATE: Trigger for arming state (from assembly to armed state) for the win C subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
10	0x0	WIN_B_UPDATE: Trigger for arming state (from assembly to armed state) for the win B subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
9	0x0	WIN_A_UPDATE: Trigger for arming state (from assembly to armed state) for the win A subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
8	0x0	GENERAL_UPDATE: Trigger for arming state (from assembly to armed state) for a subset of the triple buffered registers. This register is also known as "Update Register" 0 = DISABLE 1 = ENABLE
3	0x0	WIN_C_ACT_REQ: Window C activation request 0= no request pending/request completed. 1= activation requested/pending 0 = DISABLE 1 = ENABLE
2	0x0	WIN_B_ACT_REQ: Window B activation request 0= no request pending/request completed. 1= activation requested/pending 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	WIN_A_ACT_REQ: Window A activation request 0= no request pending/request completed. 1= activation requested/pending 0 = DISABLE 1 = ENABLE
0	0x0	GENERAL_ACT_REQ: Non-window-specific 0= no request pending/request completed. 1= activation requested/pending 0 = DISABLE 1 = ENABLE

25.6.29 DC_CMD_DISPLAY_WINDOW_HEADER_0

Display Window Header for programming display windows and their corresponding buffer start addresses.

Class: Display Window Programming Header

Offset: 0x42 | Byte Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx000xxxx)

Bit	Reset	Description
6	0x0	WINDOW_C_SELECT: Window C Select. 0= disable window C programming. 1= enable window C programming 0 = DISABLE 1 = ENABLE
5	0x0	WINDOW_B_SELECT: Window B Select. 0= disable window B programming. 1= enable window B programming 0 = DISABLE 1 = ENABLE
4	0x0	WINDOW_A_SELECT: Window A Select. 0= disable window A programming. 1= enable window A programming 0 = DISABLE 1 = ENABLE

25.6.30 DC_CMD_REG_ACT_CONTROL_0

Register activation options

Offset: 0x43 | Byte Offset: 0x10c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0x0x0x0)

Bit	Reset	Description
6	0x0	WIN_C_ACT_CNTR_SEL: Select which counter to use for window C activation 0 = VCOUNTER 1 = HCOUNTER
4	0x0	WIN_B_ACT_CNTR_SEL: Select which counter to use for window B activation 0 = VCOUNTER 1 = HCOUNTER

Bit	Reset	Description
2	0x0	WIN_A_ACT_CNTR_SEL: Select which counter to use for window A activation 0 = VCOUNTER 1 = HCOUNTER
0	0x0	GENERAL_ACT_CNTR_SEL: Select which counter to use for general activation 0 = VCOUNTER 1 = HCOUNTER

25.7 Display COM Registers

25.7.1 DC_COM_CRC_CONTROL_0

CRC Control

CRC is provided for at speed testing and diagnostic. When CRC is enabled, the CRC logic waits for the next VSync pulse or the one after that (depending on CRC_WAIT) and then it captures one frame of data at the end of display pipeline and computes the CRC value.

After one frame of data is captured, the CRC logic will stop capturing data.

When CRC_INPUT_DATA = FULL_FRAME, DISPLAY_COMMAND.DISPLAY_CTRL_MODE should be programmed to C_DISPLAY so that CRC works properly.

When CRC_INPUT_DATA = ACTIVE_DATA, it can work on both NC_DISPLAY and C_DISPLAY modes, and can work for multiple frames if CRC is checked, disabled, and re-enabled after the end of frame v-active area and before next VSYNC.

CRC logic takes 8 bits of control signals and 24-bit RGB pixel after dither and after display color (R and B) swap option.

Input [31:0] into CRC depends on CRC_INPUT_DATA. If programmed as FULL_FRAME, input data is { LVP1, LPV0, LHP2, LHP1, LHP0, VSYNC, HSYNC, ACTIVE, R[7:0], G[7:0], B[7:0]} and CRC runs over the entire frame (including blank). If programmed as ACTIVE_DATA, input data is {R[7:0], G[7:0], B[7:0]} and CRC runs only during active display area.

Offset: 0x300 | Byte Offset: 0xc00 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	CRC_ALWAYS: CRC always: calculates CRC for every following frame. Must use with CRC_INPUT_DATA = ACTIVE_DATA. If enabled, CRC_WAIT field is ignored. 0 = DISABLE 1 = ENABLE
2	0x0	CRC_INPUT_DATA: CRC input data 0= Full frame (RGB data and control) 1= Active display (Only RGB data) 0 = FULL_FRAME 1 = ACTIVE_DATA
1	0x0	CRC_WAIT: CRC Wait 0= 1 Vsync 1= 2 Vsyncs
0	0x0	CRC_ENABLE: CRC Enable 0 = DISABLE 1 = ENABLE

25.7.2 DC_COM_CRC_CHECKSUM_0

CRC Checksum

This register can be read by host after CRC logic stops capturing data.

Offset: 0x301 | Byte Offset: 0xc04 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	0x0	CRC_CHECKSUM: CRC Checksum

25.7.3 DC_COM_PIN_MISC_CONTROL_0

Pin Miscellaneous Control

Pin Miscellaneous Control

Offset: 0x31b | Byte Offset: 0xc6c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xx)

Bit	Reset	Description
2	0x0	DISP_CLOCK_OUTPUT: Display Clock (DCLK) Enable. 0= disable. 1= enable display clock to be output on LCD_DE pin (LCD_DE output select must be appropriately programmed for this to be effective) 0 = DISABLE 1 = ENABLE

25.7.4 DC_COM_PM0_CONTROL_0

PM0 Signal Control

Class: Pulse Width Modulation

PM0 signal is programmable pulse width modulation signal that can be output on several pins. The control register should be initialized once before PM0 is enabled.

> **Note:** The period is expressed as multiples of 4 times the divider value.
> The actual period - in clock cycles - is given by:
> period = (1 + PM0_CLOCK_DIVIDER) * ((PM0_PERIOD + 1) * 4)

Offset: 0x31c | Byte Offset: 0xc70 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:18	X	PM0_PERIOD: PM0 Period (4, 8, ... , 256 clock cycles)
17:4	X	PM0_CLOCK_DIVIDER: PM0 Clock Divider (1 to 16384)
1:0	X	PM0_CLOCK_SELECT: PM0 Clock Select. 0= output of shift clock divider 1= pixel clock 2= line clock 3= frame clock Notes: 1) Pixel clock, line clock, and frame clock are running only when the PW0 signal is enabled. 2) In non-continuous mode, shift clock and pixel clock run continuously, but line clock and frame clock only run while a frame is being sent.

25.7.5 DC_COM_PM0_DUTY_CYCLE_0

PM0 Duty Cycle

A counter repeatedly counts up from 0 to ((PM0_PERIOD << 2) + 3) pre-scaled cycles.

The period always starts with the output value == 1. After DUTY_CYCLE number of pre-scaled cycles, the output value is cleared for the remainder of the period. In order to output constant 0, simply set the DUTY_CYCLE to 0. To output a constant 1, set DUTY_CYCLE to be any value greater than ((PM0_PERIOD << 2) + 3).

Offset: 0x31d | Byte Offset: 0xc74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8:0	X	PM0_DUTY_CYCLE: PM0 Duty Cycle (or D) From 1/P to D/P pulse high time where P is the period.

25.7.6 DC_COM_SCRATCH_REGISTER_A_0

Scratch Register A

Class: Software Scratch Registers

Offset: 0x325 | Byte Offset: 0xc94 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH_REGISTER_A: Scratch Register A

25.7.7 DC_COM_SCRATCH_REGISTER_B_0

Scratch Register B

Offset: 0x326 | Byte Offset: 0xc98 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH_REGISTER_B: Scratch Register B

25.7.8 DC_COM_CRC_CHECKSUM_LATCHED_0

CRC Checksum latched

This register is a latched version of CRC_CHECKSUM. Latching happens at frame end.

> **Note:** CRC_INPUT_DATA needs to be set to ACTIVE_DATA if this register is used. In full frame mode, CRC is frozen two cycles after frame end due to pipelining, so only in active area mode, CRC is consistent and independent of display control mode, and can be checked continuously frame by frame.

Offset: 0x329 | Byte Offset: 0xca4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	0x0	CRC_CHECKSUM_LATCHED: CRC Checksum latched

25.7.9 DC_COM_CMU_CSC_KRR_0

CMU Color Space Conversion Matrix

R' = sat(KRR * R + KGR * G + KBR * B)

G' = sat(KRG * R + KGG * G + KBG * B)

B' = sat(KRB * R + KGB * G + KBB * B)

Coefficients are signed 1.8 fixed point, for a range of approximately [-2.0, +1.996]

Offset: 0x32a | Byte Offset: 0xca8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KRR

25.7.10 DC_COM_CMU_CSC_KGR_0

Offset: 0x32b | Byte Offset: 0xcac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KGR

25.7.11 DC_COM_CMU_CSC_KBR_0

Offset: 0x32c | Byte Offset: 0xcb0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KBR

25.7.12 DC_COM_CMU_CSC_KRG_0

Offset: 0x32d | Byte Offset: 0xcb4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KRG

25.7.13 DC_COM_CMU_CSC_KGG_0

Offset: 0x32e | Byte Offset: 0xcb8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KGG

25.7.14 DC_COM_CMU_CSC_KBG_0

Offset: 0x32f | Byte Offset: 0xcbc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KBG

25.7.15 DC_COM_CMU_CSC_KRB_0

Offset: 0x330 | Byte Offset: 0xcc0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KRB

25.7.16 DC_COM_CMU_CSC_KGB_0

Offset: 0x331 | Byte Offset: 0xcc4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KGB

25.7.17 DC_COM_CMU_CSC_KBB_0

Offset: 0x332 | Byte Offset: 0xcc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	KBB

25.7.18 DC_COM_CMU_LUT1_0

Entries are written indirectly.

Ensure CMU_ENABLE=DISABLE or display idle

For C = 0 .. 255:

C' = ungamma_function(C/255) * ((1 << NV_DISPLAY_CMU_DP_WIDTH) - 1)

Write CMU_LUT1.LUT1_ADDR = C, LUT1_DATA = C'

Write DISP_COLOR_CONTROL.CMU_ENABLE = ENABLE

Use GENERAL_ACT_REQ to activate CMU_ENABLE

CMU LUT1

Offset: 0x336 | Byte Offset: 0xcd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
27:16	X	LUT1_DATA: Readable only when LUT1_READ_EN=1
7:0	0x0	LUT1_ADDR

25.7.19 DC_COM_CMU_LUT2_0

Write sequence:

Ensure CMU_ENABLE=DISABLE or display idle

For C = 0 .. NV_DISPLAY_CMU_LUT2_DEPTH-1:

C' = gamma_function(inverse_zone_select(C)/((1 << NV_DISPLAY_CMU_DP_WIDTH) - 1)) * 255

Write CMU_LUT2.LUT2_ADDR = C, LUT2_DATA = C'

Write DISP_COLOR_CONTROL.CMU_ENABLE = ENABLE

Use GENERAL_ACT_REQ

CMU LUT2

Offset: 0x337 | Byte Offset: 0xcdc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
23:16	X	LUT2_DATA: Readable only when LUT2_READ_EN=1
9:0	0x0	LUT2_ADDR

25.8 Display DISP Registers

These registers control DISP display only; not including the DISP display window parameters.

25.8.1 DC_DISP_DISP_SIGNAL_OPTIONS0_0

Display Signal Options 0

Offset: 0x400 | Byte Offset: 0x1000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx0x0xxx000x0xxx0x0x0xxxxxxxx)

Bit	Reset	Description
26	0x0	M1_ENABLE: M1 Enable 0 = DISABLE 1 = ENABLE
24	0x0	M0_ENABLE: M0 Enable 0 = DISABLE 1 = ENABLE
20	0x0	V_PULSE3_ENABLE: V Pulse 3 Enable 0 = DISABLE 1 = ENABLE
19	0x0	V_PULSE2_ENABLE: V Pulse 2 Enable 0 = DISABLE 1 = ENABLE
18	0x0	V_PULSE1_ENABLE: V Pulse 1 Enable 0 = DISABLE 1 = ENABLE
16	0x0	V_PULSE0_ENABLE: V Pulse 0 Enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	0x0	H_PULSE2_ENABLE: H Pulse 2 Enable 0 = DISABLE 1 = ENABLE
10	0x0	H_PULSE1_ENABLE: H Pulse 1 Enable 0 = DISABLE 1 = ENABLE
8	0x0	H_PULSE0_ENABLE: H Pulse 0 Enable 0 = DISABLE 1 = ENABLE

25.8.2 DC_DISP_DISP_WIN_OPTIONS_0

Display Window Options

Offset: 0x402 │ Byte Offset: 0x1008 │ Read/Write: R/W │ Reset: 0x00000000 (0bx00xxxxxxxxxxx0xxxxxxxxxxxxxxx)

Bit	Reset	Description
30	0x0	HDMI_ENABLE: HDMI interface. The HDMI unit must also be separately enabled in its own register space in order to use HDMI functionality. 0 = DISABLE 1 = ENABLE
29	0x0	DSI_ENABLE: MIPI Display Serial Interface Enable. The DSI unit must also be separately enabled in its own register space in order to use DSI functionality. 0 = DISABLE 1 = ENABLE
16	0x0	CURSOR_ENABLE: Cursor Enable 0 = DISABLE 1 = ENABLE

25.8.3 DC_DISP_DISP_TIMING_OPTIONS_0

Display Timing Options

Class: Display Standard Timings

Programming of display timing registers must meet these restrictions:

Constraint 1: H_REF_TO_SYNC + H_SYNC_WIDTH + H_BACK_PORCH > 20.

Constraint 2: V_REF_TO_SYNC + V_SYNC_WIDTH + V_BACK_PORCH > 1.

Constraint 3: V_FRONT_PORCH + V_SYNC_WIDTH + V_BACK_PORCH > 1 (vertical blank).

Constraint 4: V_SYNC_WIDTH >= 1

H_SYNC_WIDTH >= 1

Constraint 5: V_REF_TO_SYNC >= 1

H_REF_TO_SYNC >= 0

Constraint 6: V_FRONT_PORCH >= (V_REF_TO_SYNC + 1)

H_FRONT_PORCH >= (H_REF_TO_SYNC + 1)

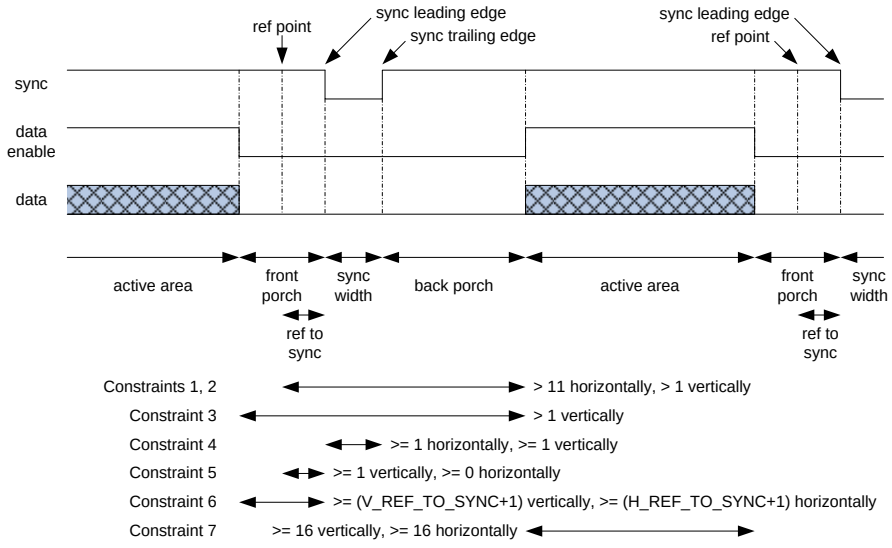
Constraint 7: H_DISP_ACTIVE >= 16

V_DISP_ACTIVE >= 16

Timing Diagram

- This diagram applies to both vertical and horizontal timing
- The back porch is the only parameter that can be negative

Figure 105: Display Timing Options Timing Diagram



This register specifies display timing options for HSYNC and VSYNC.

Offset: 0x405 │ Byte Offset: 0x1014 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	VSYNC_H_POSITION: VSYNC Horizontal Position. This parameter specifies the position where VSYNC can toggle with respect to H reference point.

25.8.4 DC_DISP_REF_TO_SYNC_0

H/V Reference to Sync

This register specifies the start position of HSYNC and VSYNC with respect to H and V reference point (line and frame start) correspondingly. The H and V reference points correspond to the time when H and V display timing counter is re-initialized to zero correspondingly.

The H reference point also determines the point where V display timing counter is incremented so this points the horizontal relationship between HSYNC and VSYNC.

Note: VSYNC's rising/falling edge is fixed at H reference point zero.

Offset: 0x406 | Byte Offset: 0x1018 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_REF_TO_SYNC: V reference to VSYNC (minimum 1 line clock)
12:0	X	H_REF_TO_SYNC: H reference to HSYNC (minimum 0 pixel clock)

25.8.5 DC_DISP_SYNC_WIDTH_0

H/V SYNC Pulse Width

This register specifies the width of HSYNC and VSYNC pulses. Check the Display Timing Options section for programming restrictions for REF_TO_SYNC.

Offset: 0x407 | Byte Offset: 0x101c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_SYNC_WIDTH: VSYNC pulse width (minimum 1 line clock)
12:0	X	H_SYNC_WIDTH: HSYNC pulse width (minimum 1 pixel clock)

25.8.6 DC_DISP_BACK_PORCH_0

H/V Back Porch

This register specifies the distance between H/V SYNC trailing edge to beginning of display active area. This is a 2's complement value, and a negative value indicates that H/V SYNC overlaps with the corresponding display active area. Check the Display Timing Options section for programming restrictions for REF_TO_SYNC.

Offset: 0x408 | Byte Offset: 0x1020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_BACK_PORCH: V back porch
12:0	X	H_BACK_PORCH: H back porch

25.8.7 DC_DISP_DISP_ACTIVE_0

H/V Display Active Width

This register specifies the width of the H/V display active area. Check the Display Timing Options section for programming restrictions for REF_TO_SYNC.

Offset: 0x409 | Byte Offset: 0x1024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_DISP_ACTIVE: V display active width (minimum 16 lines)
12:0	X	H_DISP_ACTIVE: H display active width (minimum 16 pixels)

25.8.8 DC_DISP_FRONT_PORCH_0

H/V Front Porch

This register specifies the distance between end of H/V display active area to the leading edge of the corresponding H/V SYNC.

H/V active end plus the H/V front porch value minus the H/V reference to H/VSYNC determines the H/V total (final H/V count value for the H/V display counter). Check the Display Timing Options section for programming restrictions for REF_TO_SYNC.

Offset: 0x40a │ Byte Offset: 0x1028 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_FRONT_PORCH: VSYNC front porch (minimum -=PS_=-V_REF_TO_SYNC + 1)
12:0	X	H_FRONT_PORCH: HSYNC front porch (minimum -=PS_=-H_REF_TO_SYNC + 1)

25.8.9 DC_DISP_H_PULSE0_CONTROL_0

H Pulse 0 Control

Class: Display Extended Timings

Horizontal pulse 0 is programmable pulse that repeats every line.

In NORMAL mode, this signal can have several pulses (A to D) per line with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

In ONE_CLOCK mode, this signal can have up to twice the number of pulses per line with each pulse having a width of 1 pixel clock. In this mode, the position of the one-clock pulses corresponds to the enabled Start and End positions.

Regardless of the mode, the pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

Polarity adjustment is made prior to V display qualification. This register specifies options for Horizontal pulse 0.

Offset: 0x40b │ Byte Offset: 0x102c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	H_PULSE0_LAST: H Pulse 0 Last point. 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position 6= end on Start D position 7= end on End D position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C 6 = START_D 7 = END_D
7:6	X	H_PULSE0_V_QUAL: H Pulse 0 Vertical Qualifier. 0= always running 2= run during vertical active area 3= run during vertical active plus 1 line 0 = ALWAYS 2 = VACTIVE 3 = VACTIVE1
4	X	H_PULSE0_POLARITY: H Pulse 0 Polarity. Polarity adjustment is done before the vertical qualifier is applied. 0 = HIGH 1 = LOW
3	X	H_PULSE0_MODE: H Pulse 0 Mode 0= Normal mode 1= Single-clock mode 0 = NORMAL 1 = ONE_CLOCK

25.8.10 DC_DISP_H_PULSE0_POSITION_A_0

H Pulse 0 Position A

Offset: 0x40c │ Byte Offset: 0x1030 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE0_END_A: H Pulse 0 End A (minimum -=PS_=-H_PULSE0_START_A+1)
12:0	X	H_PULSE0_START_A: H Pulse 0 Start A (minimum 0)

25.8.11 DC_DISP_H_PULSE0_POSITION_B_0

H Pulse 0 Position B

Offset: 0x40d │ Byte Offset: 0x1034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE0_END_B: H Pulse 0 End B (minimum -=PS_=-H_PULSE0_START_B+1)
12:0	X	H_PULSE0_START_B: H Pulse 0 Start B (minimum -=PS_=-H_PULSE0_END_A+1)

25.8.12 DC_DISP_H_PULSE0_POSITION_C_0

H Pulse 0 Position C

Offset: 0x40e │ Byte Offset: 0x1038 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE0_END_C: H Pulse 0 End C (minimum -=PS_=-H_PULSE0_START_C+1)
12:0	X	H_PULSE0_START_C: H Pulse 0 Start C (minimum -=PS_=-H_PULSE0_END_B+1)

25.8.13 DC_DISP_H_PULSE0_POSITION_D_0

H Pulse 0 Position D

Offset: 0x40f │ Byte Offset: 0x103c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE0_END_D: H Pulse 0 End D (minimum -=PS_=-H_PULSE0_START_D+1)
12:0	X	H_PULSE0_START_D: H Pulse 0 Start D (minimum -=PS_=-H_PULSE0_END_C+1)

25.8.14 DC_DISP_H_PULSE1_CONTROL_0

H Pulse 1 Control

Horizontal pulse 1 is programmable pulse that repeats every line.

In NORMAL mode, this signal can have several pulses (A to D) per line with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

In ONE_CLOCK mode, this signal can have up to twice the number of pulses per line with each pulse having a width of 1 pixel clock. In this mode, the position of the one-clock pulses corresponds to the enabled Start and End positions.

Regardless of the mode, the pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value), then the pulse generator will stop at the last valid position.

Polarity adjustment is made prior to V display qualification. This register specifies options for Horizontal pulse 1.

Offset: 0x410 │ Byte Offset: 0x1040 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	H_PULSE1_LAST: H Pulse 1 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position 6= end on Start D position 7= end on End D position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C 6 = START_D 7 = END_D
7:6	X	H_PULSE1_V_QUAL: H Pulse 1 Vertical Qualifier 0= always running 2= run during vertical active area 3= run during vertical active plus 1 line 0 = ALWAYS 2 = VACTIVE 3 = VACTIVE1
4	X	H_PULSE1_POLARITY: H Pulse 1 Polarity. Polarity adjustment is done before the vertical qualifier is applied 0 = HIGH 1 = LOW
3	X	H_PULSE1_MODE: H Pulse 1 Mode 0= Normal mode 1= Single-clock mode 0 = NORMAL 1 = ONE_CLOCK

25.8.15 DC_DISP_H_PULSE1_POSITION_A_0

H Pulse 1 Position A

Offset: 0x411 │ Byte Offset: 0x1044 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE1_END_A: H Pulse 1 End A (minimum -=PS_=-H_PULSE1_START_A+1)
12:0	X	H_PULSE1_START_A: H Pulse 1 Start A (minimum 0)

25.8.16 DC_DISP_H_PULSE1_POSITION_B_0

H Pulse 1 Position B

Offset: 0x412 | Byte Offset: 0x1048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE1_END_B: H Pulse 1 End B (minimum -=PS_=-H_PULSE1_START_B+1)
12:0	X	H_PULSE1_START_B: H Pulse 1 Start B (minimum -=PS_=-H_PULSE1_END_A+1)

25.8.17 DC_DISP_H_PULSE1_POSITION_C_0

H Pulse 1 Position C

Offset: 0x413 | Byte Offset: 0x104c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE1_END_C: H Pulse 1 End C (minimum -=PS_=-H_PULSE1_START_C+1)
12:0	X	H_PULSE1_START_C: H Pulse 1 Start C (minimum -=PS_=-H_PULSE1_END_B+1)

25.8.18 DC_DISP_H_PULSE1_POSITION_D_0

H Pulse 1 Position D

Offset: 0x414 | Byte Offset: 0x1050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE1_END_D: H Pulse 1 End D (minimum -=PS_=-H_PULSE1_START_D+1)
12:0	X	H_PULSE1_START_D: H Pulse 1 Start D (minimum -=PS_=-H_PULSE1_END_C+1)

25.8.19 DC_DISP_H_PULSE2_CONTROL_0

H Pulse 2 Control

Horizontal pulse 2 is a programmable pulse that repeats every line.

In NORMAL mode, this signal can have several pulses (A to D) per line with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

In ONE_CLOCK mode this signal can have up to twice the number of pulses per line with each pulse having a width of 1 pixel clock. In this mode, the position of the one-clock pulses corresponds to the enabled Start and End positions.

Regardless of the mode, the pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc. So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value), then the pulse generator will stop at the last valid position. Polarity adjustment is made prior to V display qualification. This register specifies options for Horizontal pulse 2.

Offset: 0x415 │ Byte Offset: 0x1054 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	H_PULSE2_LAST: H Pulse 2 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position 6= end on Start D position 7= end on End D position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C 6 = START_D 7 = END_D
7:6	X	H_PULSE2_V_QUAL: H Pulse 2 Vertical Qualifier 0= always running 2= run during vertical active area 3= run during vertical active plus 1 line 0 = ALWAYS 2 = VACTIVE 3 = VACTIVE1
4	X	H_PULSE2_POLARITY: H Pulse 2 Polarity. Polarity adjustment is done before the vertical qualifier is applied 0 = HIGH 1 = LOW
3	X	H_PULSE2_MODE: H Pulse 2 Mode. 0= Normal mode 1= Single-clock mode 0 = NORMAL 1 = ONE_CLOCK

25.8.20 DC_DISP_H_PULSE2_POSITION_A_0

H Pulse 2 Position A

Offset: 0x416 │ Byte Offset: 0x1058 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE2_END_A: H Pulse 2 End A (minimum -=PS_=-H_PULSE2_START_A+1)
12:0	X	H_PULSE2_START_A: H Pulse 2 Start A (minimum 0)

25.8.21 DC_DISP_H_PULSE2_POSITION_B_0

H Pulse 2 Position B

Offset: 0x417 | Byte Offset: 0x105c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE2_END_B: H Pulse 2 End B (minimum -=PS_=-H_PULSE2_START_B+1)
12:0	X	H_PULSE2_START_B: H Pulse 2 Start B (minimum -=PS_=-H_PULSE2_END_A+1)

25.8.22 DC_DISP_H_PULSE2_POSITION_C_0

H Pulse 2 Position C

Offset: 0x418 | Byte Offset: 0x1060 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE2_END_C: H Pulse 2 End C (minimum -=PS_=-H_PULSE2_START_C+1)
12:0	X	H_PULSE2_START_C: H Pulse 2 Start C (minimum -=PS_=-H_PULSE2_END_B+1)

25.8.23 DC_DISP_H_PULSE2_POSITION_D_0

H Pulse 2 Position D

Offset: 0x419 | Byte Offset: 0x1064 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	H_PULSE2_END_D: H Pulse 2 End D (minimum -=PS_=-H_PULSE2_START_D+1)
12:0	X	H_PULSE2_START_D: H Pulse 2 Start D (minimum -=PS_=-H_PULSE2_END_C+1)

25.8.24 DC_DISP_V_PULSE0_CONTROL_0

V Pulse 0 Control

Vertical pulse 0 is a programmable pulse that repeats every frame.

This signal can have several pulses (A to C) per frame with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc. In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

The pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc. So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 0.

Offset: 0x41a │ Byte Offset: 0x1068 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	0x0	V_PULSE0_H_POSITION: V Pulse 0 Horizontal Position. This parameter specifies the position where V Pulse 0 can toggle with respect to H reference point.
11:8	X	V_PULSE0_LAST: V Pulse 0 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C
7:6	X	V_PULSE0_DELAY: V Pulse 0 Delay 0= no delay 1= 1-line delay 2= 2-line delay 3= reserved 0 = NODELAY 1 = DELAY1 2 = DELAY2
4	X	V_PULSE0_POLARITY: V Pulse 0 Polarity 0 = HIGH 1 = LOW

25.8.25 DC_DISP_V_PULSE0_POSITION_A_0

V Pulse 0 Position A

Offset: 0x41b │ Byte Offset: 0x106c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE0_END_A: V Pulse 0 End A (minimum -=PS_=-V_PULSE0_START_A+1)
12:0	X	V_PULSE0_START_A: V Pulse 0 Start A (minimum 0)

25.8.26 DC_DISP_V_PULSE0_POSITION_B_0

V Pulse 0 Position B

Offset: 0x41c │ Byte Offset: 0x1070 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE0_END_B: V Pulse 0 End B (minimum -=PS_=-V_PULSE0_START_B+1)
12:0	X	V_PULSE0_START_B: V Pulse 0 Start B (minimum -=PS_=-V_PULSE0_END_A+1)

25.8.27 DC_DISP_V_PULSE0_POSITION_C_0

V Pulse 0 Position C

Offset: 0x41d │ Byte Offset: 0x1074 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE0_END_C: V Pulse 0 End C (minimum -=PS_=-V_PULSE0_START_C+1)
12:0	X	V_PULSE0_START_C: V Pulse 0 Start C (minimum -=PS_=-V_PULSE0_END_B+1)

25.8.28 DC_DISP_V_PULSE1_CONTROL_0

V pulse 1 Control

Vertical pulse 1 is a programmable pulse that repeats every frame. This signal can have several pulses (A to C) per frame with programmable width as defined by the pairs of start and end positions. The pulses must not overlap and must occur in sequence: pulse A, then pulse B, etc.

In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

The pulse generator processes the pairs of start and end position sequentially in the order of: Start A, End A, Start B, End B, etc. So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 1.

Offset: 0x41e │ Byte Offset: 0x1078 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxx0xxxx)

Bit	Reset	Description
28:16	0x0	V_PULSE1_H_POSITION: V Pulse 1 Horizontal Position. This parameter specifies the position where V Pulse 1 can toggle with respect to H reference point.
11:8	X	V_PULSE1_LAST: V pulse 1 Last point 0= end on Start A position 1= end on End A position 2= end on Start B position 3= end on End B position 4= end on Start C position 5= end on End C position others= reserved 0 = START_A 1 = END_A 2 = START_B 3 = END_B 4 = START_C 5 = END_C

Bit	Reset	Description
7:6	X	V_PULSE1_DELAY: V pulse 1 Delay 0= no delay 1= 1-line delay 2= 2-line delay 3= reserved 0 = NODELAY 1 = DELAY1 2 = DELAY2
4	X	V_PULSE1_POLARITY: V pulse 1 Polarity 0 = HIGH 1 = LOW

25.8.29 DC_DISP_V_PULSE1_POSITION_A_0

V Pulse 1 Position A

Offset: 0x41f │ Byte Offset: 0x107c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE1_END_A: V Pulse 1 End A (minimum -=PS_=-V_PULSE1_START_A+1)
12:0	X	V_PULSE1_START_A: V Pulse 1 Start A (minimum 0)

25.8.30 DC_DISP_V_PULSE1_POSITION_B_0

V Pulse 1 Position B

Offset: 0x420 │ Byte Offset: 0x1080 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE1_END_B: V Pulse 1 End B (minimum -=PS_=-V_PULSE1_START_B+1)
12:0	X	V_PULSE1_START_B: V Pulse 1 Start B (minimum -=PS_=-V_PULSE1_END_A+1)

25.8.31 DC_DISP_V_PULSE1_POSITION_C_0

V Pulse 1 Position C

Offset: 0x421 │ Byte Offset: 0x1084 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE1_END_C: V Pulse 1 End C (minimum -=PS_=-V_PULSE1_START_C+1)
12:0	X	V_PULSE1_START_C: V Pulse 1 Start C (minimum -=PS_=-V_PULSE1_END_B+1)

25.8.32 DC_DISP_V_PULSE2_CONTROL_0

V Pulse 2 Control

Vertical pulse 2 is a programmable pulse that repeats every frame.

This signal can have one pulse (A) per frame with programmable width as defined by the pair of start and end positions.

In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 2.

Offset: 0x422 │ Byte Offset: 0x1088 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxx0xxxx)

Bit	Reset	Description
28:16	0x0	V_PULSE2_H_POSITION: V Pulse 2 Horizontal Position. This parameter specifies the position where V Pulse 2 can toggle with respect to the H reference point.
8	X	V_PULSE2_LAST: V pulse 2 Last point 0= end on Start A position 1= end on End A position others= reserved 0 = START_A 1 = END_A
4	0x0	V_PULSE2_POLARITY: V pulse 2 Polarity 0 = HIGH 1 = LOW

25.8.33 DC_DISP_V_PULSE2_POSITION_A_0

V Pulse 2 Position A

Offset: 0x423 │ Byte Offset: 0x108c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxx0xxxx)

Bit	Reset	Description
28:16	X	V_PULSE2_END_A: V Pulse 2 End A (minimum -=PS_=-V_PULSE2_START_A+1)
12:0	X	V_PULSE2_START_A: V Pulse 2 Start A (minimum 0)

25.8.34 DC_DISP_V_PULSE3_CONTROL_0

V Pulse 3 Control

Vertical pulse 3 is a programmable pulse that repeats every frame.

This signal can have one pulse (A) per frame with programmable width as defined by the pair of start and end positions.

In this case, the Enable field must be set to one of the End position. If the Enable field is set to one of the Start position then the pulse generator will stop as if the Enable field is set to the previous End position. If the Enable field is set to the Start A position, then no pulse is generated.

So these start/end positions should be programmed in increasing order. If any of the positions are programmed in non-increasing order (has invalid value) then the pulse generator will stop at the last valid position.

This register specifies options for Vertical pulse 3.

Offset: 0x424 │ Byte Offset: 0x1090 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0000000000000xxxxxxxxxxx0xxxx)

Bit	Reset	Description
28:16	0x0	V_PULSE3_H_POSITION: V Pulse 3 Horizontal Position. This parameter specifies the position where V Pulse 3 can toggle with respect to H reference point.
8	X	V_PULSE3_LAST: V pulse 3 Last point 0= end on Start A position 1= end on End A position others= reserved 0 = START_A 1 = END_A
4	0x0	V_PULSE3_POLARITY: V pulse 3 Polarity 0 = HIGH 1 = LOW

25.8.35 DC_DISP_V_PULSE3_POSITION_A_0

V Pulse 3 Position A

Offset: 0x425 │ Byte Offset: 0x1094 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	V_PULSE3_END_A: V Pulse 3 End A (minimum -=PS_=-V_PULSE3_START_A+1)
12:0	X	V_PULSE3_START_A: V Pulse 3 Start A (minimum 0)

25.8.36 DC_DISP_DISP_CLOCK_CONTROL_0

Display Clock Control

The shift clock divider is used to divide root clock for display controller module to generate internal shift clock for shifting data to the display. Output of this divider is typically used to generate the external shift clock which is sent to the display (SC0 and/or SC1) except for 1-pixel/1-clock parallel display.

The output of this divider is also used to generate Programmable Pulse (PP) signal. For 1-pixel/1-clock parallel display, SC0 and SC1 are generated using the output of pixel clock divider which can be set to 1, 2, or 4 for 1-pixel/1-clock parallel display.

The reason pixel clock divider 2 and 4 are allowed for 1-pixel/1-clock parallel display interface is so that the clock that generates PP can be generated with 2x or 4x higher frequency than pixel clock and therefore can produce higher resolution PP pulse positions. For all cases of parallel display, SC0 and SC1 can be further divided by 1, 2, or 4.

Class: Display Interface Settings

This register controls generation of shift clock to the display and internal pixel clock. Internal display pipeline runs with pixel clock and processes 1 pixel per clock.

Offset: 0x42e │ Byte Offset: 0x10b8 │ Read/Write: R/W │ Reset: 0x00000006 (0bxxxxxxxxxxxxxxxxxxxx000000000110)

Bit	Reset	Description
11:8	0x0	PIXEL_CLK_DIVIDER: Pixel Clock Divider 0000= divide by 1 0001= divide by 1.5 0010= divide by 2 0011= divide by 3 0100= divide by 4 0101= divide by 6 0110= divide by 8 0111= divide by 9 1000= divide by 12 1001= divide by 16 1010= divide by 18 1011= divide by 24 1100= divide by 13 other= reserved 0 = PCD1 1 = PCD1H 2 = PCD2 3 = PCD3 4 = PCD4 5 = PCD6 6 = PCD8 7 = PCD9 8 = PCD12 9 = PCD16 10 = PCD18 11 = PCD24 12 = PCD13

Bit	Reset	Description
7:0	0x6	SHIFT_CLK_DIVIDER: Shift Clock Divider. 0 = divide by 1 1 = divide by 1.5 2 = divide by 2 3 = divide by 2.5 4 = divide by 3 : : : : 254 = divide by 128 255 = divide by 128.5 Pixel clock divider is used to divide output of internal shift clock divider to generate internal pixel clock which is used to clock the internal horizontal and vertical counters. This divider also determines the output format for parallel interface, serial interface, and LCD SPI interface in conjunction with Display Data Format parameter. For 1-pixel/1-clock parallel display interface, valid settings are PCD1, PCD2, and PCD4. Note that the main reason to use PCD2 and PCD4 is to get higher frequency PP clock because the PP clock is always generated from the output of shift clock divider. For non 1-pixel/1-clock parallel display interface, valid settings are, PCD1H (2-pixel/3-clock), PCD2 (1-pixel/2-clock), and PCD3 (1-pixel/3-clock). For 1-channel serial display interface, valid settings are PCD3 (3-bpp 1-ch), PCD4 (3-bpp 1-ch), PCD6 (6-bpp 1-ch), PCD9 (9-bpp 1-ch), PCD12 (12-bpp 1-ch), PCD16 (16-bpp 1-ch), PCD18 (18-bpp 1-ch). For 2-channel serial display interface, valid settings are PCD2 (3-bpp 2-ch), PCD3 (6-bpp 2-ch), PCD6 (12-bpp 2-ch), PCD8 (16-bpp 2-ch), PCD9 (18-bpp 2-ch). For 3-channel serial display interface, valid settings are PCD1 (3-bpp 3-ch), PCD2 (6-bpp 3-ch), PCD3 (9-bpp 3-ch), PCD4 (12-bpp 3-ch), PCD6 (18-bpp 3-ch). For LCD SPI interface, valid settings are PCD12 (B4G4R4), PCD16 (B5G6R5), PCD18 (B6G6R6), PCD24 (B8G8R8), PCD8 (B5G6R5 with data/command bit), PCD6 (B5G6R5 with data/command start byte - depending on data/command bit), PCD4 (P8 for spi8), PCD9 (B5G6R5 with chip select deassertion at 8-bit boundary, spi16x2), PCD3 (P8 for spidc), PCD2 (B5G6R5 with data/command bit and chip select de-assertion at 9-bit boundary, spi16x2dc), and PCD13 (spi12p2, no chip select de-assertion between pairs of pixels).

25.8.37 DC_DISP_DISP_INTERFACE_CONTROL_0

Display Interface Control

This register specifies display interface options.

Note: Only the PCD1 setting is allowed.
Only default settings are allowed.

Offset: 0x42f │ Byte Offset: 0x10bc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00xxxx0000)

Bit	Reset	Description
9	0x0	DISP_DATA_ORDER: Display Data Order. This is effective only for 1-pixel/2-clock 16-/18-/24- bit parallel interface 0= Red pixel is output in the first clock and blue pixel is output in the second cycle 1= Blue pixel is output in the first clock cycle and red pixel is output in the second clock cycle 0 = RED_BLUE 1 = BLUE_RED

Bit	Reset	Description
8	0x0	DISP_DATA_ALIGNMENT: Display Data Alignment. This is effective for parallel display data format and the associated Initialization Sequence (IS). 0= Output data is MSB-aligned. For 1-pixel/1-clock parallel display, the output data ordering is the same regardless of display Base Color Size. For 1-pixel/1-clock parallel display data alignment is optimized for 18-bpp so the 24-bit data ordering is: LD[5:0] are blue data bits 7-2 LD[11:6] are green data bits 7-2 LD[17:12] are red data bits 7-2 LD[19:18] are blue data bits 1-0 LD[21:20] are green data bits 1-0 LD[23:22] are red data bits 1-0 Note that LD18 to LD23 signals are multiplexed with control pins (see Pin Output Select definition) 1= Output data is LSB-aligned. For 1-pixel/1-clock parallel display the output data ordering is determined by display Base Color Size. For 1-pixel/1-clock parallel display data alignment is optimized for 24-bpp as follows: LD[7:0] are blue data bits 7-0 LD[15:8] are green data bits 7-0 LD[23:16] are red data bits 7-0 Note that LD18 to LD23 signals are multiplexed with control pins (see Pin Output Select definition) 0 = MSB 1 = LSB
3:0	0x0	DISP_DATA_FORMAT: Display Data Format Pixel Clock Divider is used together with this parameter to determine the exact display data format. 0 = DF1P1C : 0= 1-pixel/1-clock up to 24-bit parallel 1 = DF1P2C24B : 1= 1-pixel/2-clock 24-bit parallel 2 = DF1P2C18B : 2= 1-pixel/2-clock 18-bit parallel or 2-pixel/3-clock 12-bit parallel or 1-pixel/3-clock 18-bit parallel NOTE: for 2-pixel/3-clock 12-bit parallel, the horizontal display active time must be even number of pixels. 3 = DF1P2C16B : 3= 1-pixel/2-clock 16-bit parallel 4 = DF1S : 4= 1-channel serial NOTE: 1-/2-/3-channel serial display interface supported is a low-voltage differential serial interface. 5 = DF2S : 5= 2-channel serial 6 = DF3S : 6= 3-channel serial 7 = DFSPI : 7= SPI serial 8 = DF1P3C24B : 8= 1-pixel/3-clock 24-bit parallel 9 = DF2P1C18B : 9= 2-pixel/1-clock 18-bit parallel 10 = DFDUAL1P1C18B: 10= 1-pixel/1-clock 18-bit parallel (used for dual output)

25.8.38 DC_DISP_DISP_COLOR_CONTROL_0

Display Color Control

Offset: 0x430 │ Byte Offset: 0x10c0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxx0000xxx0x0x0xx00xxxxxxxx0000)

Bit	Reset	Description
27	0x0	LCD_MD3: LCD Mode 3 signal 0 = LOW 1 = HIGH
26	0x0	LCD_MD2: LCD Mode 2 signal 0 = LOW 1 = HIGH

Bit	Reset	Description
25	0x0	LCD_MD1: LCD Mode 1 signal 0 = LOW 1 = HIGH
24	0x0	LCD_MD0: LCD Mode 0 signal 0 = LOW 1 = HIGH
20	DISABLE	CMU_ENABLE: MD0-3 signals are general purpose mode signals that can be output in various pins (see Pin Output Select) to configure the display device. These bits are effective at start of frame. Typically these can be programmed in shadow register which takes effect on the next frame. 0 = DISABLE 1 = ENABLE
18	0x0	NON_BASE_COLOR: Non Base Color 0= zeros 1= ones
17	X	BLANK_COLOR: Blank Color 0= zeros 1= ones Non Base Color applies to least significant color bits which are not part of base color and it has higher priority over Border Color but lower priority over Blank color.
16	0x0	DISP_COLOR_SWAP: Display Color Swap 0= RGB (normal) 1= BGR (red-blue reverse) 0 = RGB 1 = BGR
13:12	0x0	ORD_DITHER_ROTATION: Ordered Dither Frame Rotation. This parameter specifies the rotation frequency of the dither matrix in terms of number of frames. If programmed to 0, there is no dither matrix rotation. If programmed to N, where N is larger than 0, the dither matrix is rotated clockwise every N frame.
9:8	X	DITHER_CONTROL: Dither Control 00= dither disabled 01= reserved 10= ordered dither 11= error-diffusion dither (maximum line width is 1280) Design Note: initial dither matrix (where d is 2 dither bits) d=00 d=01 d=10 d=11 --------- ---------- --------- --------- 0 0 1 0 0 1 0 1 --------- --------- --------- --------- 0 0 0 0 1 0 1 1 --------- ---- Note: 0 in the matrix specifies no addition to base color 1 in the matrix specifies incrementation of base color (with saturation) 0 = DISABLE 2 = ORDERED 3 = ERRDIFF

Bit	Reset	Description
3:0	0x0	BASE_COLOR_SIZE: Display Base Color Size. This parameter determines the number of bits per color after dither. 0= 6 bits 1= 1 bit 2= 2 bits 3= 3 bits 4= 4 bits 5= 5 bits 6= 5 bits for R,B and 6 bits for G 7= 3 bits for R,G and 2 bits for B 8= 8 bits, this also forces dither to be disabled. This setting can be used to output 24-bit data in 1-pixel/clock parallel display data format. 0 = BASE666 1 = BASE111 2 = BASE222 3 = BASE333 4 = BASE444 5 = BASE555 6 = BASE565 7 = BASE332 8 = BASE888

25.8.39 DC_DISP_BORDER_COLOR_0

Border Color

Border Color defines the color of areas within the active display area which are outside the defined active windows. This is 24-bit color which is applied after blending.

Offset: 0x435 │ Byte Offset: 0x10d4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	BORDER_COLOR_B: Blue Border Color
15:8	X	BORDER_COLOR_G: Green Border Color
7:0	X	BORDER_COLOR_R: Red Border Color

25.8.40 DC_DISP_COLOR_KEY0_LOWER_0

Color Key 0 Lower Value

Color Key 0 and Color Key 1

Two ranges of color key are defined and they are common for all windows because it is expected that typically only one window will have the color key enabled. Because there are two sets of color key, it is possible to have 2 windows each using one color key set.

Usage of this color key is described in the Display Color Key and Blending class.

Offset: 0x436 │ Byte Offset: 0x10d8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	COLOR_KEY0_L_B: Color Key 0 Blue (U) Lower value
15:8	X	COLOR_KEY0_L_G: Color Key 0 Green (Y) Lower value

Bit	Reset	Description
7:0	X	COLOR_KEY0_L_R: Color Key 0 Red (V) Lower value

25.8.41 DC_DISP_COLOR_KEY0_UPPER_0

Color Key 0 Upper Value

Offset: 0x437 | Byte Offset: 0x10dc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	COLOR_KEY0_U_B: Color Key 0 Blue (U) Upper value
15:8	X	COLOR_KEY0_U_G: Color Key 0 Green (Y) Upper value
7:0	X	COLOR_KEY0_U_R: Color Key 0 Red (V) Upper value

25.8.42 DC_DISP_COLOR_KEY1_LOWER_0

Color Key 1 Lower Value

Offset: 0x438 | Byte Offset: 0x10e0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	COLOR_KEY1_L_B: Color Key 1 Blue (U) Lower value
15:8	X	COLOR_KEY1_L_G: Color Key 1 Green (Y) Lower value
7:0	X	COLOR_KEY1_L_R: Color Key 1 Red (V) Lower value

25.8.43 DC_DISP_COLOR_KEY1_UPPER_0

Color Key 1 Upper Value

Offset: 0x439 | Byte Offset: 0x10e4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	COLOR_KEY1_U_B: Color Key 1 Blue (U) Upper value
15:8	X	COLOR_KEY1_U_G: Color Key 1 Green (Y) Upper value
7:0	X	COLOR_KEY1_U_R: Color Key 1 Red (V) Upper value

25.8.44 DC_DISP_CURSOR_FOREGROUND_0

Cursor Foreground Color

Class: Hardware Cursor

Hardware cursor is supported for 32x32 or for 64x64 2-bpp cursor.

Cursor start address is aligned to 1 KB boundary. All cursor registers except for cursor foreground and background colors are triple buffered.

GENERAL_UPDATE controls ASSEMBLY->ARM latching, GENERAL_ACT_REQ controls ARM->ACTIVE latching.

Cursor scaling and flipping are not implemented so this must be done by software if needed. Cursor H/V positions are signed number with respect to one of the display windows or with respect to upper left position of display active area as specified by cursor clipping parameter which also determines cursor clipping boundary. If cursor position is with respect to one of the display window and the corresponding display window is disabled then cursor will also be disabled.

Offset: 0x43c │ Byte Offset: 0x10f0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CURSOR_FOREGROUND_B: Cursor Blue Foreground Color
15:8	X	CURSOR_FOREGROUND_G: Cursor Green Foreground Color
7:0	X	CURSOR_FOREGROUND_R: Cursor Red Foreground Color

25.8.45 DC_DISP_CURSOR_BACKGROUND_0

Cursor Background Color

Offset: 0x43d │ Byte Offset: 0x10f4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	CURSOR_BACKGROUND_B: Cursor Blue Background Color
15:8	X	CURSOR_BACKGROUND_G: Cursor Green Background Color
7:0	X	CURSOR_BACKGROUND_R: Cursor Red Background Color

25.8.46 DC_DISP_CURSOR_START_ADDR_0

Cursor Start Address

Offset: 0x43e │ Byte Offset: 0x10f8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:28	X	CURSOR_CLIPPING: Cursor Clipping Select 0 = DISPLAY : 00= display 1 = WA : 01= window A 2 = WB : 10= window B 3 = WC : 11= window C
25: 24	X	CURSOR_SIZE: Cursor Size 0 = C32X32 1 = C64X64 2 = C128X128 3 = C256X256

Bit	Reset	Description
21:0	X	CURSOR_START_ADDR: Cursor Start Address bits 25:10

25.8.47 DC_DISP_CURSOR_START_ADDR_NS_0

Shadow of Cursor Start Address

Offset: 0x43f │ Byte Offset: 0x10fc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:28	X	CURSOR_CLIPPING_NS: Cursor Clipping Select 0 = DISPLAY : 00= display 1 = WA : 01= window A 2 = WB : 10= window B 3 = WC : 11= window C
25:24	X	CURSOR_SIZE_NS: Cursor Size 0 = C32X32 1 = C64X64 2 = C128X128 3 = C256X256
21:0	X	CURSOR_START_ADDR_NS: Cursor Start Address bits 25:10

25.8.48 DC_DISP_CURSOR_POSITION_0

Cursor Position

Cursor position is with respect to top-left corner of display active area, window A, window B, or window C as specified in the cursor clipping parameter.

Offset: 0x440 │ Byte Offset: 0x1100 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:16	X	V_CURSOR_POSITION: V cursor position (signed)
13:0	X	H_CURSOR_POSITION: H cursor position (signed)

25.8.49 DC_DISP_CURSOR_POSITION_NS_0

Shadow of Cursor Position

Offset: 0x441 │ Byte Offset: 0x1104 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:16	X	V_CURSOR_POSITION_NS: V cursor position (signed)
13:0	X	H_CURSOR_POSITION_NS: H cursor position (signed)

25.8.50 DC_DISP_DC_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register

Note: The FIFO timing aspects of this register are no longer supported, but are retained for software compatibility. The clock enable fields of this register control the second level clock gating for the MC side of the MCCIF.

A '1' written to the override filed will result in that clock reverting to the legacy mode of operation (where the msc*clk is on whenever the client clock is enabled.

Offset: 0x480 | Byte Offset: 0x1200 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx00xxxxxxxxxxx0000)

Bit	Reset	Description
17	0x0	DC_RCLK_OVERRIDE
16	0x0	DC_WCLK_OVERRIDE
3	DISABLE	DC_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	DC_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	DC_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	DC_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

25.8.51 DC_DISP_MCCIF_DISPLAY0A_HYST_0

Memory Client Hysteresis Control Register

Note: Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 0x481 | Byte Offset: 0x1204 | Read/Write: R/W | Reset: 0xcf401f1f (0b11001111010000000001111100011111)

Bit	Reset	Description
31	ENABLE	CSR_DISPLAY0A2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CSR_DISPLAY0A2MC_HYST_REQ_TH
27:24	0xf	CSR_DISPLAY0A2MC_HYST_TM
23:16	0x38	CSR_DISPLAY0A2MC_DHYST_TH
15:8	0x10	CSR_DISPLAY0A2MC_DHYST_TM

Bit	Reset	Description
7:0	0x58	CSR_DISPLAY0A2MC_HYST_REQ_TM

25.8.52 DC_DISP_MCCIF_DISPLAY0B_HYST_0

Memory Client Hysteresis Control Register

Note: Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 0x482 | Byte Offset: 0x1208 | Read/Write: R/W | Reset: 0xcf401f1f (0b11001111010000000001111100011111)

Bit	Reset	Description
31	ENABLE	CSR_DISPLAY0B2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CSR_DISPLAY0B2MC_HYST_REQ_TH
27:24	0xf	CSR_DISPLAY0B2MC_HYST_TM
23:16	0x40	CSR_DISPLAY0B2MC_DHYST_TH
15:8	0x1f	CSR_DISPLAY0B2MC_DHYST_TM
7:0	0x1f	CSR_DISPLAY0B2MC_HYST_REQ_TM

25.8.53 DC_DISP_MCCIF_DISPLAY0C_HYST_0

Memory Client Hysteresis Control Register

Note: Hysteresis is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 0x483 | Byte Offset: 0x120c | Read/Write: R/W | Reset: 0xcf401f1f (0b11001111010000000001111100011111)

Bit	Reset	Description
31	ENABLE	CSR_DISPLAY0C2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CSR_DISPLAY0C2MC_HYST_REQ_TH
27:24	0xf	CSR_DISPLAY0C2MC_HYST_TM
23:16	0x40	CSR_DISPLAY0C2MC_DHYST_TH
15:8	0x1f	CSR_DISPLAY0C2MC_DHYST_TM
7:0	0x1f	CSR_DISPLAY0C2MC_HYST_REQ_TM

25.8.54 DC_DISP_DISP_MISC_CONTROL_0

Miscellaneous Controls

Offset: 0x4c1 | Byte Offset: 0x1304 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx10)

Bit	Reset	Description
1	0x1	UF_LINE_FLUSH: Enable underflow line flush as opposed to end-of-frame flush. underflow line flush 0 = DISABLE; 1 = ENABLE
0	0x0	PHASE_SHIFT_2P1C18B: Enable phase shift for 2P1C format phase shift SC0/SC1 will be delayed for one pixel clock cycle. In 2P1C format, data will hold for 2 pixel clocks, so either choice should work 0 = DISABLE 1 = ENABLE

25.8.55 DC_DISP_SD_CONTROL_0

The Smart Dimmer function takes advantage of the fact that a perceived pixel brightness in an LCD depends on both the pixel brightness value and the backlight intensity to reduce the backlight intensity to save power. Statistics are gathered on the current video frame and an "enhancement" is applied to subsequent frames that increases the pixel brightness value and reduces the backlight brightness to give an overall image intensity that is mostly the same as before.

Smart Dimmer Control

Offset: 0x4c2 | Byte Offset: 0x1308 | Read/Write: R/W | Reset: 0x00004000 (0bxxx0xxxxxxxxxxxx0100000000000000)

Bit	Reset	Description
28	VSYNC	SD_FRAME_PROC_CONTROL: when to run per-frame processing. VSYNC is legacy behavior. 0 = VSYNC 1 = VPULSE2
15	DISABLE	SMOOTH_K_ENABLE: When enable, maximum raw K change per frame is limited to SMOOTH_K_INCR 0 = DISABLE 1 = ENABLE
14	ENABLE	SOFT_CLIPPING_ENABLE: When enabled, enhancement gain (K) is reduced for pixels above SOFT_CLIPPING_THRESHOLD level to avoid saturation (clipping). 0 = DISABLE 1 = ENABLE
13	DISABLE	SD_WINDOW_ENABLE: When enabled, constrain histogram (and therefore backlight) to a rectangular subset of display. The rectangle has upper/left corner described by SD_WINDOW_POSITION, and width/height by SD_WINDOW_SIZE. Useful for when display has regions that should not influence SD, such as letterbox borders, etc. Enhancement is always performed on whole display, regardless. DISABLE: SD histogram done over whole display ENABLE: SD histogram done over rectangular subset given by SD_WINDOW* registers 0 = DISABLE 1 = ENABLE
12	DISABLE	K_LIMIT_ENABLE: When enabled, Max. K is taken from K_LIMIT register rather than computed from AGGRESSIVENESS. DISABLE: K is limited by AGGRESSIVENESS ENABLE: K is limited by K_LIMIT register. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x0	SD_CORRECTION_MODE: Determines which K values are used to modify the pixel values. MANUAL : The K values in the SD_MAN_K_VALUES register are used to modify pixel values. 0 = AUTO_CORRECT : AUTO_CORRECT : SD Hardware computed K values are used to 1 = MANUAL
10	0x0	SD_ONE_SHOT: Enables the Smart Dimmer function for one frame only. on which the SD statistics are gathered. NOTE: The SD_ENABLE field (see above) must be set to ONE_SHOT in order to use this function. 0 = DISABLE : Automatically cleared to DISABLE at the end of the frame 1 = ENABLE
9:8	0x0	HW_UPDATE_DLY: Determines the delay - in video frames - of the update of the hardware enhancement value that is applied to the pixels. This is useful for allowing the software some time to update the backlight control, when the control must be sent via side-band control packets or by some other means of control that incurs a sizeable delay. Being able to delay the hardware update ensures that the modification of the pixels occurs as nearly simultaneously with the update of the backlight as possible. Value Description 0 No delay - pixels modified immediately 1 New enhancement value delayed by 1 frame. 2 New enhancement value delayed by 2 frames. 3 New enhancement value delayed by 3 frames.
7:5	0x0	AGGRESSIVENESS: The "aggressiveness" level of the Smart Dimmer algorithm. Higher aggressiveness levels result in higher power savings at the potential expense of image quality. The number programmed determines how many highlight pixels will be allowed to exceed the maximum representable brightness value and be clipped to that value. It also determines the maximum allowed enhancement value (k) applied to the pixel brightness. AGGRESSIVENESS Description % pixels Max. k value crushed value 0 Essentially off 0% 1.00 1 Highest quality < 5% 1.10 2 Higher quality < 10% 1.15 3 Balanced < 15% 1.20 4 Higher battery life < 20% 1.25 5 Highest battery life < 25% 1.50
4:3	0x0	BIN_WIDTH: Width of the Histogram bins, in quantization levels. EIGHT = 8 levels per bin. Bins span range from 0 to 255 0 = ONE : ONE = 1 level per bin. Bins span range from 224 to 255 1 = TWO : TWO = 2 levels per bin. Bins span range from 192 to 255 2 = FOUR : FOUR = 4 levels per bin. Bins span range from 128 to 255 3 = EIGHT
2	0x0	USE_VID_LUMA: Use Video Luminance control of luminance: Luminance = MAX(R, G, B) ENABLE = use "video" luminance, which is determined by the coefficients in the SD_CSC_COEFFS register See the SD_CSC_COEFFS register for details. 0 = DISABLE : DISABLE = use Hue Saturation Value (HSV) version 1 = ENABLE
1:0	0x0	SD_ENABLE: Enables the Smart Dimmer Function ONE_SHOT = SD function enabled, but statistics gathering limited to the next frame only. 0 = DISABLE : DISABLE = SD function disabled. 1 = ENABLE : ENABLE = SD function enabled and operational. 2 = ONE_SHOT

25.8.56 DC_DISP_SD_CSC_COEFF_0

Luminance calculation coefficients used to convert the red, green, and blue color components into a luminance value. The conversion is performed according to the following equation: Luminance = (R * R_COEFF + G * G_COEFF + B * B_COEFF) >> 4 It is suggested that the values of the coefficients be programmed as shown below, though user-defined color spaces are also accommodated. Color Space R_COEFF G_COEFF B_COEFF ITU-R Bt601 5 9 2 ITU-R Bt709 3 12 1. The coefficients

do not have to be particularly accurate, hence their low precision, and the coefficients used are open to experimentation to obtain the best results.

Offset: 0x4c3 | Byte Offset: 0x130c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx0000xxxx0000xxxx0000xxxx)

Bit	Reset	Description
23:20	0x0	B_COEFF: Blue coefficient for luminance calculation
15:12	0x0	G_COEFF: Green coefficient for luminance calculation
7:4	0x0	R_COEFF: Red coefficient for luminance calculation

25.8.57 DC_DISP_SD_LUT_0

Enhancement value (k) Look Up Table. Each LUT entry contains the value of k for each of the three color components. Since the value of k for the color components must be the reciprocal of the hardware-computed value of k, and the hardware value is guaranteed to be less than or equal to 1, the values in the LUT represent the fractional part of k, with an implied 1 to the left of the decimal place. For example, if the programmed value of R_LUT was 64 (01000000 in binary), then the actual value of k generated would be 1.01000000 in binary or 1.25 in decimal. To program a default, linear response into the LUT, use the following code as a guide: for (i = 0; i < 9; i++) { t = (4096 / (8 + i)) - 256; if (t > 255) t = 255; R_LUT[i] = t; G_LUT[i] = t; B_LUT[i] = t; } For other non-linear response curves (for example, to take display gamma into consideration), this code will have to be modified.

This is an array of 9 identical register entries; the register fields below apply to each entry.

Offset: 0x4c4..0x4cc | Byte Offset: 0x1310..0x1330 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	B_LUT: Blue Enhancement value (k) Look Up Table
15:8	0x0	G_LUT: Green Enhancement value (k) Look Up Table
7:0	0x0	R_LUT: Red Enhancement value (k) Look Up Table

25.8.58 DC_DISP_SD_FLICKER_CONTROL_0

Flicker Reduction Control Register

The flicker control prevents rapid and frequent changes in the enhancement value.

Offset: 0x4cd | Byte Offset: 0x1334 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	THRESHOLD: The amount by which the currently calculated enhancement value must deviate from the currently active enhancement value for it to increment the TIME_LIMIT counter.
7:0	0x0	TIME_LIMIT: Length of time - in frames - that the enhancement value must deviate from the current value by more than THRESHOLD, before the enhancement value changes.

25.8.59 DC_DISP_SD_PIXEL_COUNT_0

Status / debug register showing the total number of active pixels

Offset: 0x4ce │ Byte Offset: 0x1338 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	NUM_PIXELS: in the preceding output frame. Expressed as a quantity of 256 pixels. In other words, a 640 x 480 image has 307200 pixels. The value in this register would be 307200 / 256 = 1200

25.8.60 DC_DISP_SD_HISTOGRAM_0

Status/debug registers showing the gathered histogram data. Each register contains 4 histogram bins, for a total of 8 x 4 = 32 bins. Each bin has been approximately scaled to the number of pixels in the image so that a single quantization step in a bin represents a fraction of between 1/256 and 1/128 of the total number of pixels in the image.

This is an array of 8 identical register entries; the register fields below apply to each entry.

Offset: 0x4cf..0x4d6 │ Byte Offset: 0x133c..0x1358 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	BIN_3
23:16	X	BIN_2
15:8	X	BIN_1
7:0	X	BIN_0

25.8.61 DC_DISP_SD_BL_PARAMETERS_0

Backlight response parameters. Defines the parameters for the backlight temporal response model.

Offset: 0x4d7 │ Byte Offset: 0x135c │ Read/Write: R/W │ Reset: 0x00000400 (0bxxxxxxxx00000000xxxxx10000000000)

Bit	Reset	Description
23:16	0x0	STEP: Determines the instantaneous portion of the target value of enhancement that is applied. 0 = 0% : response is entirely exponential and determined by TIME_CONSTANT 128 = 50% : response will instantly step up by 50% and will then be exponential. 255 = 100% : response is entirely instantaneous. TIME_CONSTANT has no effect.
10:0	0x400	TIME_CONSTANT: The time constant for the response curve. This value represents the fraction by which the value of enhancement value approaches the target value each frame. Example values are shown below: 0 : The value will never reach the target (infinite TC) 512 : The next value will be halfway between the current value and the target value. 1024 : The next value will be 100% of the target value. In other words - an instantaneous response.

25.8.62 DC_DISP_SD_BL_TF_0

Backlight Transfer Function

Each register contains 4 points on the Transfer Function curve that defines how the backlight output changes with respect to the control input. Each point defines a value at the vertex of a 16 segment line. The 17th point is defined to be the maximum value (it is assumed 100% control == 100% light output).

This is an array of 4 identical register entries; the register fields below apply to each entry.

Offset: 0x4d8..0x4db │ Byte Offset: 0x1360..0x136c │ Read/Write: R/W │Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	POINT_3
23:16	X	POINT_2
15:8	X	POINT_1
7:0	X	POINT_0

25.8.63 DC_DISP_SD_BL_CONTROL_0

Backlight Control Register

Offset: 0x4dc │ Byte Offset: 0x1370 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
15:8	RO	X	BRIGHTNESS: Backlight brightness modification value. This value is determined by the hardware according to all the other control registers and the image content. The amount by which the backlight should be modified is given as a fraction with 0 representing that the backlight should be off and 255 representing no change in the backlight intensity. Other values vary linearly between these two extremes. New BL control = (Old BL control * BRIGHTNESS) / 255
1:0	RW	0x0	BL_MODE: Control Mode: and adjust the backlight brightness itself. PWM_AUTO : Hardware will adjust the backlight PWM control signal directly using the value in BRIGHTNESS. * OTHER VALUES ARE RESERVED FOR FUTURE USE * 0 = MANUAL : MANUAL : Hardware makes no BL corrections. Software must read the BRIGHTNESS field 1 = PWM_AUTO

25.8.64 DC_DISP_SD_HW_K_VALUES_0

Hardware-computed values of K for each color component. These values are used to modify the pixel values when CORRECTION_MODE in the SD_CONTROL register is set to AUTO_CORRECT. The numerical format is a fractional 10-bit number. Since we only ever want to make the pixel values bigger, there is an implied integer "1" not present in the register value. Only the fractional part is returned. Also in Tegra 4 devices, only the top 7 bits of the 10-bit K word have any meaning. The bottom 3 bits are always 0 and are reserved for future expansion. This register is read only.

Offset: 0x4dd │ Byte Offset: 0x1374 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:20	X	HW_K_BLUE: Value of K for blue pixels.
19:10	X	HW_K_GREEN: Value of K for green pixels.
9:0	X	HW_K_RED: Value of K for red pixels.

25.8.65 DC_DISP_SD_MAN_K_VALUES_0

Manual values of K for each color component. These values are used to modify the pixel values when CORRECTION_MODE in the SD_CONTROL register is set to MANUAL. The numerical format is a fractional 10-bit number. Since we only ever want to make the pixel values bigger, there is an implied integer "1" not present in the register value. Only the fractional part can be programmed. Therefore, K values can only be programmed from 1.0 to slightly less than 2.0 .Also note that only the top 7 bits of the 10-bit K word have any effect in the hardware. The bottom 3 bits are reserved for future expansion.

Offset: 0x4de | Byte Offset: 0x1378 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:20	X	MAN_K_BLUE: Value of K for blue pixels.
19:10	X	MAN_K_GREEN: Value of K for green pixels.
9:0	X	MAN_K_RED: Value of K for red pixels.

25.8.66 DC_DISP_SD_K_LIMIT_0

Offset: 0x4df | Byte Offset: 0x137c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	K_LIMIT: When K_LIMIT_ENABLE=ENABLE, limits raw K independently of AGGRESSIVENESS. Currently, only the top 8 bits are effective. The bottom 2 are reserved for future expansion.

25.8.67 DC_DISP_SD_WINDOW_POSITION_0

SD Window Position

Offset: 0x4e0 | Byte Offset: 0x1380 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	SD_WIN_V_POSITION: SD window vertical position. This is specified with respect to the top edge of active display area.
12:0	X	SD_WIN_H_POSITION: SD window horizontal position (pixels). This is specified with respect to the left edge of active display area.

25.8.68 DC_DISP_SD_WINDOW_SIZE_0

SD Window Size

Offset: 0x4e1 | Byte Offset: 0x1384 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	SD_WIN_V_SIZE: SD window vertical height (lines).
12:0	X	SD_WIN_H_SIZE: SD window horizontal width (pixels).

25.8.69 DC_DISP_SD_SOFT_CLIPPING_0

SD soft clipping parameters. Software programs both threshold and a reciprocal thereof.

Offset: 0x4e2 | Byte Offset: 0x1388 | Read/Write: R/W | Reset: 0x02000080 (0b00000010000000000xxxxxxx10000000)

Bit	Reset	Description
31:16	0x200	SOFT_CLIPPING_RECIP: Reciprocal of inverse threshold. Compute as RECIP = int(64*1024 * 1.0 / (256 - THRESHOLD)). For example, for THRESHOLD=128, RECIP = int(64*1024 * 1.0 / (256 - 128)) = 512.
7:0	0x80	SOFT_CLIPPING_THRESHOLD: Threshold at which pixel enhancement gain is reduced.

25.8.70 DC_DISP_SD_SMOOTH_K_0

Offset: 0x4e3 | Byte Offset: 0x138c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13:0	X	SMOOTH_K_INCR: When SMOOTH_K_ENABLE=1, the raw K is changed at most by SMOOTH_K_INCR per frame. 8.6 fixed-point fraction, with resolution 1/64. Currently, only the top 12 bits are effective. The bottom 2 are reserved for future expansion, so the effective resolution is 1/16.

25.8.71 DC_DISP_CURSOR_UNDERFLOW_CTRL_0

Offset: 0x4eb | Byte Offset: 0x13ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx1xxxxxx0)

Bit	Reset	Description
7	0x0	CURSOR_UFLOW_CYA: 0 = DISABLE 1 = ENABLE
0	0x0	CURSOR_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

25.8.72 DC_DISP_BLEND_CURSOR_CONTROL_0

Offset: 0x4f1 | Byte Offset: 0x13c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxx0xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24	0x0	CURSOR_MODE_SELECT: 0 = LEGACY 1 = NORMAL

25.8.73 DC_DISP_DVFS_CURSOR_CONTROL_0

Programmable threshold to deassert for cursor. Default value is 3 lines ahead of cursor active.

Offset: 0x4f2 | Byte Offset: 0x13c8 | Read/Write: R/W | Reset: 0x00000003 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000011)

Bit	Reset	Description
7:0	0x3	CURSOR_DVFS_THRESHOLD

25.8.74 DC_DISP_CURSOR_UFLOW_DBG_PIXEL_0

Offset: 0x4f3 | Byte Offset: 0x13cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CURSOR_UFLOW_DBG_PIXEL

25.8.75 DC_DISP_CURSOR_SPOOLUP_CONTROL_0

Programmable spool up for cursor. If spoolup count > hc_vactive_start, it will be clamped to zero.

Offset: 0x4f4 | Byte Offset: 0x13d0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000001)

Bit	Reset	Description
7:0	0x1	CURSOR_SPOOLUP_START

25.8.76 DC_DISP_DISPLAY_CLK_GATE_OVERRIDE_0

Offset: 0x4f5 | Byte Offset: 0x13d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	CMU_CLK_GATE_OVERRIDE: Disable clock-gating of the CMU module. 0 = DISABLE 1 = ENABLE
0	0x0	CURSOR_CLK_GATE_OVERRIDE: Disable clock-gating of cursor memfetch/control modules 0 = DISABLE 1 = ENABLE

25.8.77 DC_DISP_DISPLAY_DBG_TIMING_0

Offset: 0x4f6 | Byte Offset: 0x13d8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	H_BLANK
28:16	X	H_COUNT
15	X	V_BLANK
12:0	X	V_COUNT

25.8.78 DC_DISP_DISPLAY_SPARE0_0

Offset: 0x4f7 | Byte Offset: 0x13dc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH_0

25.8.79 DC_DISP_DISPLAY_SPARE1_0

Offset: 0x4f8 | Byte Offset: 0x13e0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH_1

25.9 Window A (WINC_A) Registers

These registers control window A parameters.

> **Note:** There are three copies of these registers for windows A, B, and C. Windows A, B, and C support different features

Window A: color palette, digital vibrance

Window B: color palette, digital vibrance, color space conversion, horizontal/vertical filtering

Window C: color palette, digital vibrance, color space conversion, horizontal filtering. The registers under DC_WINC are usually not shadowed.

25.9.1 DC_WINC_A_COLOR_PALETTE_0

This is used for palletized data format (color depth of 8 bpp or less) or for gamma correction for non-palletized data formats (color depth of more than 8 bpp).

Each window has its own color palette which consists of three 256x8 register files which can be written by the host and indexed (read) by the window.

For palletized data formats less than 8 bpp, the pixel data is aligned to least significant bits of the palette index (address), and the remaining upper bits are filled with the corresponding bits of the Palette Color Extension. For example, for 4-bpp mode, the pixel data occupies bits 3:0 of the palette index, and bits 7:4 of the palette index are set to bits 7:4 of the Palette Color Extension.

Note that a host read is assumed to be not needed - software can cache the color palette in system memory.

This is an array of 256 identical register entries; the register fields below apply to each entry.

Window A Color Palette

Offset: 0x500..0x5ff │ Byte Offset: 0x1400..0x17fc │ Read/Write: WO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	A_COLOR_PALETTE_B: Blue Color Palette
15:8	X	A_COLOR_PALETTE_G: Green Color Palette
7:0	X	A_COLOR_PALETTE_R: Red Color Palette

25.9.2 DC_WINC_A_PALETTE_COLOR_EXT_0

Palette extension for 1-bpp, 2-bpp, and 4-bpp. These bits provide the upper most significant bits for indexing the color palette.

Window A Palette Color Extension

Offset: 0x600 │ Byte Offset: 0x1800 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:1	X	A_PALETTE_COLOR_EXT: Window A Palette Color Extension. Bits 7:1 are used for 1-bpp mode, bits 7:2 are used for 2-bpp mode, and bits 7:4 are used for 4-bpp mode

25.9.3 DC_WINC_A_H_FILTER_P00_0

Horizontal Scaling Filter Coefficients

Horizontal scaling filter is a 6-tap filter with 4-bit positional phase.

- Coefficients 0 and 5 are 3-bit signed values ranging from -4 to 3.
- Coefficients 1 and 4 are 5-bit signed values ranging from -16 to 15.
- Coefficients 2 and 3 are 8-bit unsigned values ranging from 0 to 128.
- Coefficient 0 is the multiplier for the earliest pixel (P0) in the group of 6 pixels and
- Coefficient 5 is the multiplier for the latest pixel (P5) in the group. The output pixel positional phase is defined as centered in P2 if the positional phase is 0 or proportionally in between P2 and P3 if the positional phase is larger than 0.

The sum of all coefficients for each phase should be 128 typically, and software should never program the sum of all coefficients for a phase to be more than 128.

For each horizontal positional phase, the 6 filter coefficients require 32 register bits.

Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Window A Horizontal Filter phase 00

Offset: 0x601 | Byte Offset: 0x1804 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P00C5: Phase 00 coefficient 5 (typically 0)
28:24	X	A_H_FILTER_P00C4: Phase 00 coefficient 4 (typically 0)
23:16	X	A_H_FILTER_P00C3: Phase 00 coefficient 3 (typically 0)
15:8	X	A_H_FILTER_P00C2: Phase 00 coefficient 2 (typically 128)
7:3	X	A_H_FILTER_P00C1: Phase 00 coefficient 1 (typically 0)
2:0	X	A_H_FILTER_P00C0: Phase 00 coefficient 0 (typically 0)

25.9.4 DC_WINC_A_H_FILTER_P01_0

Window A Horizontal Filter phase 01

Offset: 0x602 | Byte Offset: 0x1808 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P01C5: Phase 01 coefficient 5 (typically 1)
28:24	X	A_H_FILTER_P01C4: Phase 01 coefficient 4 (typically -2)
23:16	X	A_H_FILTER_P01C3: Phase 01 coefficient 3 (typically 8)
15:8	X	A_H_FILTER_P01C2: Phase 01 coefficient 2 (typically 124)
7:3	X	A_H_FILTER_P01C1: Phase 01 coefficient 1 (typically -4)
2:0	X	A_H_FILTER_P01C0: Phase 01 coefficient 0 (typically 1)

25.9.5 DC_WINC_A_H_FILTER_P02_0

Window A Horizontal Filter phase 02

Offset: 0x603 | Byte Offset: 0x180c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P02C5: Phase 02 coefficient 5 (typically 1)
28:24	X	A_H_FILTER_P02C4: Phase 02 coefficient 4 (typically -5)
23:16	X	A_H_FILTER_P02C3: Phase 02 coefficient 3 (typically 17)
15:8	X	A_H_FILTER_P02C2: Phase 02 coefficient 2 (typically 122)
7:3	X	A_H_FILTER_P02C1: Phase 02 coefficient 1 (typically -8)
2:0	X	A_H_FILTER_P02C0: Phase 02 coefficient 0 (typically 1)

25.9.6 DC_WINC_A_H_FILTER_P03_0

Window A Horizontal Filter phase 03

Offset: 0x604 | Byte Offset: 0x1810 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P03C5: Phase 03 coefficient 5 (typically 2)
28:24	X	A_H_FILTER_P03C4: Phase 03 coefficient 4 (typically -7)
23:16	X	A_H_FILTER_P03C3: Phase 03 coefficient 3 (typically 27)
15:8	X	A_H_FILTER_P03C2: Phase 03 coefficient 2 (typically 115)
7:3	X	A_H_FILTER_P03C1: Phase 03 coefficient 1 (typically -11)
2:0	X	A_H_FILTER_P03C0: Phase 03 coefficient 0 (typically 2)

25.9.7 DC_WINC_A_H_FILTER_P04_0

Window A Horizontal Filter phase 04

Offset: 0x605 | Byte Offset: 0x1814 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P04C5: Phase 04 coefficient 5 (typically 2)
28:24	X	A_H_FILTER_P04C4: Phase 04 coefficient 4 (typically -9)
23:16	X	A_H_FILTER_P04C3: Phase 04 coefficient 3 (typically 37)
15:8	X	A_H_FILTER_P04C2: Phase 04 coefficient 2 (typically 109)
7:3	X	A_H_FILTER_P04C1: Phase 04 coefficient 1 (typically -13)
2:0	X	A_H_FILTER_P04C0: Phase 04 coefficient 0 (typically 2)

25.9.8 DC_WINC_A_H_FILTER_P05_0

Window A Horizontal Filter phase 05

Offset: 0x606 | Byte Offset: 0x1818 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P05C5: Phase 05 coefficient 5 (typically 2)
28:24	X	A_H_FILTER_P05C4: Phase 05 coefficient 4 (typically -11)
23:16	X	A_H_FILTER_P05C3: Phase 05 coefficient 3 (typically 47)
15:8	X	A_H_FILTER_P05C2: Phase 05 coefficient 2 (typically 102)
7:3	X	A_H_FILTER_P05C1: Phase 05 coefficient 1 (typically -15)
2:0	X	A_H_FILTER_P05C0: Phase 05 coefficient 0 (typically 3)

25.9.9 DC_WINC_A_H_FILTER_P06_0

Window A Horizontal Filter phase 06

Offset: 0x607 | Byte Offset: 0x181c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P06C5: Phase 06 coefficient 5 (typically 3)
28:24	X	A_H_FILTER_P06C4: Phase 06 coefficient 4 (typically -13)
23:16	X	A_H_FILTER_P06C3: Phase 06 coefficient 3 (typically 56)
15:8	X	A_H_FILTER_P06C2: Phase 06 coefficient 2 (typically 94)
7:3	X	A_H_FILTER_P06C1: Phase 06 coefficient 1 (typically -15)
2:0	X	A_H_FILTER_P06C0: Phase 06 coefficient 0 (typically 3)

25.9.10 DC_WINC_A_H_FILTER_P07_0

Window A Horizontal Filter phase 07

Offset: 0x608 | Byte Offset: 0x1820 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P07C5: Phase 07 coefficient 5 (typically 3)
28:24	X	A_H_FILTER_P07C4: Phase 07 coefficient 4 (typically -14)
23:16	X	A_H_FILTER_P07C3: Phase 07 coefficient 3 (typically 67)
15:8	X	A_H_FILTER_P07C2: Phase 07 coefficient 2 (typically 85)
7:3	X	A_H_FILTER_P07C1: Phase 07 coefficient 1 (typically -16)
2:0	X	A_H_FILTER_P07C0: Phase 07 coefficient 0 (typically 3)

25.9.11 DC_WINC_A_H_FILTER_P08_0

Window A Horizontal Filter phase 08

Offset: 0x609 | Byte Offset: 0x1824 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P08C5: Phase 08 coefficient 5 (typically 3)
28:24	X	A_H_FILTER_P08C4: Phase 08 coefficient 4 (typically -15)
23:16	X	A_H_FILTER_P08C3: Phase 08 coefficient 3 (typically 76)
15:8	X	A_H_FILTER_P08C2: Phase 08 coefficient 2 (typically 76)
7:3	X	A_H_FILTER_P08C1: Phase 08 coefficient 1 (typically -15)
2:0	X	A_H_FILTER_P08C0: Phase 08 coefficient 0 (typically 3)

25.9.12 DC_WINC_A_H_FILTER_P09_0

Window A Horizontal Filter phase 09

Offset: 0x60a | Byte Offset: 0x1828 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P09C5: Phase 09 coefficient 5 (typically 3)
28:24	X	A_H_FILTER_P09C4: Phase 09 coefficient 4 (typically -16)
23:16	X	A_H_FILTER_P09C3: Phase 09 coefficient 3 (typically 85)
15:8	X	A_H_FILTER_P09C2: Phase 09 coefficient 2 (typically 67)
7:3	X	A_H_FILTER_P09C1: Phase 09 coefficient 1 (typically -14)
2:0	X	A_H_FILTER_P09C0: Phase 09 coefficient 0 (typically 3)

25.9.13 DC_WINC_A_H_FILTER_P0A_0

Window A Horizontal Filter phase 0A

Offset: 0x60b | Byte Offset: 0x182c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P0AC5: Phase 0A coefficient 5 (typically 3)
28:24	X	A_H_FILTER_P0AC4: Phase 0A coefficient 4 (typically -15)
23:16	X	A_H_FILTER_P0AC3: Phase 0A coefficient 3 (typically 94)
15:8	X	A_H_FILTER_P0AC2: Phase 0A coefficient 2 (typically 56)
7:3	X	A_H_FILTER_P0AC1: Phase 0A coefficient 1 (typically -13)
2:0	X	A_H_FILTER_P0AC0: Phase 0A coefficient 0 (typically 3)

25.9.14 DC_WINC_A_H_FILTER_P0B_0

Window A Horizontal Filter phase 0B

Offset: 0x60c | Byte Offset: 0x1830 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P0BC5: Phase 0B coefficient 5 (typically 3)
28:24	X	A_H_FILTER_P0BC4: Phase 0B coefficient 4 (typically -15)
23:16	X	A_H_FILTER_P0BC3: Phase 0B coefficient 3 (typically 102)
15:8	X	A_H_FILTER_P0BC2: Phase 0B coefficient 2 (typically 47)
7:3	X	A_H_FILTER_P0BC1: Phase 0B coefficient 1 (typically -11)
2:0	X	A_H_FILTER_P0BC0: Phase 0B coefficient 0 (typically 2)

25.9.15 DC_WINC_A_H_FILTER_P0C_0

Window A Horizontal Filter phase 0C

Offset: 0x60d │ Byte Offset: 0x1834 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P0CC5: Phase 0C coefficient 5 (typically 2)
28:24	X	A_H_FILTER_P0CC4: Phase 0C coefficient 4 (typically -13)
23:16	X	A_H_FILTER_P0CC3: Phase 0C coefficient 3 (typically 109)
15:8	X	A_H_FILTER_P0CC2: Phase 0C coefficient 2 (typically 37)
7:3	X	A_H_FILTER_P0CC1: Phase 0C coefficient 1 (typically -9)
2:0	X	A_H_FILTER_P0CC0: Phase 0C coefficient 0 (typically 2)

25.9.16 DC_WINC_A_H_FILTER_P0D_0

Window A Horizontal Filter phase 0D

Offset: 0x60e │ Byte Offset: 0x1838 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P0DC5: Phase 0D coefficient 5 (typically 2)
28:24	X	A_H_FILTER_P0DC4: Phase 0D coefficient 4 (typically -11)
23:16	X	A_H_FILTER_P0DC3: Phase 0D coefficient 3 (typically 115)
15:8	X	A_H_FILTER_P0DC2: Phase 0D coefficient 2 (typically 27)
7:3	X	A_H_FILTER_P0DC1: Phase 0D coefficient 1 (typically -7)
2:0	X	A_H_FILTER_P0DC0: Phase 0D coefficient 0 (typically 2)

25.9.17 DC_WINC_A_H_FILTER_P0E_0

Window A Horizontal Filter phase 0E

Offset: 0x60f │ Byte Offset: 0x183c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P0EC5: Phase 0E coefficient 5 (typically 1)
28:24	X	A_H_FILTER_P0EC4: Phase 0E coefficient 4 (typically -8)
23:16	X	A_H_FILTER_P0EC3: Phase 0E coefficient 3 (typically 122)
15:8	X	A_H_FILTER_P0EC2: Phase 0E coefficient 2 (typically 17)
7:3	X	A_H_FILTER_P0EC1: Phase 0E coefficient 1 (typically -5)
2:0	X	A_H_FILTER_P0EC0: Phase 0E coefficient 0 (typically 1)

25.9.18 DC_WINC_A_H_FILTER_P0F_0

Window A Horizontal Filter phase 0F

Offset: 0x610 | Byte Offset: 0x1840 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	A_H_FILTER_P0FC5: Phase 0F coefficient 5 (typically 1)
28:24	X	A_H_FILTER_P0FC4: Phase 0F coefficient 4 (typically -4)
23:16	X	A_H_FILTER_P0FC3: Phase 0F coefficient 3 (typically 124)
15:8	X	A_H_FILTER_P0FC2: Phase 0F coefficient 2 (typically 8)
7:3	X	A_H_FILTER_P0FC1: Phase 0F coefficient 1 (typically -2)
2:0	X	A_H_FILTER_P0FC0: Phase 0F coefficient 0 (typically 1)

25.9.19 DC_WINC_A_CSC_YOF_0

Color Space Conversion Coefficients

The CSC can be used for YUV to RGB conversion with brightness and hue/saturation control. The CSC can only be enabled for window A controlled by CSC_ENABLE register bits.

For Y color, the Y offset is applied first and saturation (clipping) is performed immediately after the Y offset is applied.

R = sat(KYRGB * sat(Y + YOF) + KUR * U + KVR * V)

G = sat(KYRGB * sat(Y + YOF) + KUG * U + KVG * V)

B = sat(KYRGB * sat(Y + YOF) + KUB * U + KVB * V)

Saturation and rounding is performed in the range of 0 to 255 for the above equations.

Typical values are:

YOF = -16.000, KYRGB = 1.1644

KUR = 0.0000, KVR = 1.5960

KUG = -0.3918, KVG = -0.8130

KUB = 2.0172, KVB = 0.0000

KUR and KVB are typically 0.0000 but they may be programmed non-zero for hue rotation.

The CSC can also take RGB input, in which case YOF, KVB, KUG, and KUR should be programmed to 0, and KYRGB will be forced to 0 by the hardware for generating R and B. KYRGB will not be forced to 0 for generating G. KVR, KYRGB, and KUB can be programmed to 1.0 or used as gain control for R, G, B, correspondingly.

Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Window A CSC Y Offset

Offset: 0x611 | Byte Offset: 0x1844 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_CSC_YOF: Y Offset in s.7.0 format

25.9.20 DC_WINC_A_CSC_KYRGB_0

Window A CSC Y Coefficient (gain) for RGB

Offset: 0x612 | Byte Offset: 0x1848 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	A_CSC_KYRGB: Y Gain for R, G, B colors in 2.8 format

25.9.21 DC_WINC_A_CSC_KUR_0

Window A CSC U coefficient for R

Offset: 0x613 | Byte Offset: 0x184c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	A_CSC_KUR: U coefficients for R in s.2.8 format

25.9.22 DC_WINC_A_CSC_KVR_0

Window A CSC V coefficient for R

Offset: 0x614 | Byte Offset: 0x1850 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	A_CSC_KVR: V coefficients for R in s.2.8 format

25.9.23 DC_WINC_A_CSC_KUG_0

Window A CSC U coefficient for G

Offset: 0x615 | Byte Offset: 0x1854 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	A_CSC_KUG: U coefficients for G in s.1.8 format

25.9.24 DC_WINC_A_CSC_KVG_0

Window A CSC V coefficient for G

Offset: 0x616 | Byte Offset: 0x1858 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	A_CSC_KVG: V coefficients for G in s.1.8 format

25.9.25 DC_WINC_A_CSC_KUB_0

Window A CSC U coefficient for B

Offset: 0x617 | Byte Offset: 0x185c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	A_CSC_KUB: U coefficients for B in s.2.8 format

25.9.26 DC_WINC_A_CSC_KVB_0

Window A CSC V coefficient for B

Offset: 0x618 | Byte Offset: 0x1860 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	A_CSC_KVB: V coefficients for B in s.2.8 format

25.9.27 DC_WINC_A_V_FILTER_P00_0

Vertical Scaling Filter Coefficients

The vertical scaling filter is a 2-tap filter with 4-bit positional phase. Coefficients 0 and 1 are 8-bit unsigned values ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2 pixels, and coefficient 1 is the multiplier for the later pixel (P1) in the group. The output pixel positional phase is defined as centered in P0 if the positional phase is 0 or proportionally in between P0 and P1 if the positional phase is larger than 0.

The sum of all coefficients for each phase should be 128 typically. Therefore coefficient 1 can be calculated from (1 - coefficient 0) and only coefficient 0 is programmed. For each vertical positional phase, the filter coefficient requires 8 register bits. Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Window A Vertical Filter phase 00

Offset: 0x619 | Byte Offset: 0x1864 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P00C0: Phase 00 coefficient 0 (typically 128)

25.9.28 DC_WINC_A_V_FILTER_P01_0

Window A Vertical Filter phase 01

Offset: 0x61a | Byte Offset: 0x1868 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P01C0: Phase 01 coefficient 0 (typically 120)

25.9.29 DC_WINC_A_V_FILTER_P02_0

Window A Vertical Filter phase 02

Offset: 0x61b | Byte Offset: 0x186c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P02C0: Phase 02 coefficient 0 (typically 112)

25.9.30 DC_WINC_A_V_FILTER_P03_0

Window A Vertical Filter phase 03

Offset: 0x61c | Byte Offset: 0x1870 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P03C0: Phase 03 coefficient 0 (typically 104)

25.9.31 DC_WINC_A_V_FILTER_P04_0

Window A Vertical Filter phase 04

Offset: 0x61d | Byte Offset: 0x1874 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P04C0: Phase 04 coefficient 0 (typically 96)

25.9.32 DC_WINC_A_V_FILTER_P05_0

Window A Vertical Filter phase 05

Offset: 0x61e | Byte Offset: 0x1878 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P05C0: Phase 05 coefficient 0 (typically 88)

25.9.33 DC_WINC_A_V_FILTER_P06_0

Window A Vertical Filter phase 06

Offset: 0x61f | Byte Offset: 0x187c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P06C0: Phase 06 coefficient 0 (typically 80)

25.9.34 DC_WINC_A_V_FILTER_P07_0

Window A Vertical Filter phase 07

Offset: 0x620 | Byte Offset: 0x1880 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P07C0: Phase 07 coefficient 0 (typically 72)

25.9.35 DC_WINC_A_V_FILTER_P08_0

Window A Vertical Filter phase 08

Offset: 0x621 | Byte Offset: 0x1884 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P08C0: Phase 08 coefficient 0 (typically 64)

25.9.36 DC_WINC_A_V_FILTER_P09_0

Window A Vertical Filter phase 09

Offset: 0x622 | Byte Offset: 0x1888 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P09C0: Phase 09 coefficient 0 (typically 56)

25.9.37 DC_WINC_A_V_FILTER_P0A_0

Window A Vertical Filter phase 0A

Offset: 0x623 | Byte Offset: 0x188c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P0AC0: Phase 0A coefficient 0 (typically 48)

25.9.38 DC_WINC_A_V_FILTER_P0B_0

Window A Vertical Filter phase 0B

Offset: 0x624 | Byte Offset: 0x1890 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P0BC0: Phase 0B coefficient 0 (typically 40)

25.9.39 DC_WINC_A_V_FILTER_P0C_0

Window A Vertical Filter phase 0C

Offset: 0x625 | Byte Offset: 0x1894 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P0CC0: Phase 0C coefficient 0 (typically 32)

25.9.40 DC_WINC_A_V_FILTER_P0D_0

Window A Vertical Filter phase 0D

Offset: 0x626 | Byte Offset: 0x1898 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P0DC0: Phase 0D coefficient 0 (typically 24)

25.9.41 DC_WINC_A_V_FILTER_P0E_0

Window A Vertical Filter phase 0E

Offset: 0x627 | Byte Offset: 0x189c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P0EC0: Phase 0E coefficient 0 (typically 16)

25.9.42 DC_WINC_A_V_FILTER_P0F_0

Window A Vertical Filter phase 0F

Offset: 0x628 | Byte Offset: 0x18a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	A_V_FILTER_P0FC0: Phase 0F coefficient 0 (typically 8)

The registers under DC_WIN are double buffered.

25.9.43 DC_WIN_A_WIN_OPTIONS_0

Window A Options

Class: Display Window Settings

Display Window A parameters

Offset: 0x700 | Byte Offset: 0x1c00 | Read/Write: R/W | Reset: 0x00000000 (0bx0xxxxxxxxxxxx0x0xxxxxxxx0xxxx)

Bit	Reset	Description
30	0x0	A_WIN_ENABLE: Window A Window enable 0 = DISABLE 1 = ENABLE
22	X	A_YUV_RANGE_EXPAND: Window A Enable range expansion in the cases where RANGEREDFRM is 1 from mpd. Formula: Y = clip((Y-128)*2 + 128); Cb = clip((Cb-128)*2 + 128); Cr = clip((Cr-128)*2 + 128); where clip() function clips between 0 and 255. 0 = DISABLE 1 = ENABLE
20	X	A_DV_ENABLE: Window A Digital Vibrance Enable 0 = DISABLE 1 = ENABLE
18	X	A_CSC_ENABLE: Window A Color Space Conversion Enable. This controls the color space conversion and should be enabled for YCbCr/YUV color modes for conversion to B8G8R8 and for hue and saturation control. This can also be used for gain control for RGB color modes. 0 = DISABLE 1 = ENABLE
16	0x0	A_CP_ENABLE: Window A Color Palette Enable. This controls the color palette and should be enabled for palletized color modes. For non-palletized color modes, the color palette can be enabled for gamma correction. 0 = DISABLE 1 = ENABLE
14	0x0	A_V_FILTER_UV_ALIGN: Window A V Filter UV Alignment. This is effective only when vertical scaling filter is enabled and only on these formats YCbCr420P, YUV420P, YCbCr422R, YUV422R, YCbCr422RA YUV422RA. When UV alignment is enabled, the chroma components are aligned to the even number of luma component lines. When disabled the chroma components are aligned to half a pixel below the corresponding even number of luma component lines. It is usually disabled unless the incoming video stream specifically indicates otherwise. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	X	A_V_FILTER_OPTIMIZE: Window A V Filter Optimization. This is effective only when vertical scaling filter is enabled. This can be used (enabled) to temporarily disable the vertical scaling filter when the vertical scaling DDA fraction is zero. In this case the next line is not fetched from memory to save bandwidth and power. This feature cannot be used in 420P/422R/422RA formats. 0 = DISABLE 1 = ENABLE
10	X	A_V_FILTER_ENABLE: Window A V Filter Enable. This controls V scaling filter and is effective only for non-palletized color modes. If V filter is disabled, only one line is read from memory for each output line. NOTE: This feature is not supported,. This bit is ignored and is a don't care. 0 = DISABLE 1 = ENABLE
8	X	A_H_FILTER_ENABLE: Window A H Filter Enable. This controls H scaling filter and is effective only for non-palletized color modes. 0 = DISABLE 1 = ENABLE
6	X	A_COLOR_EXPAND: Window A 12/15/16/18-to-24 bpp color expansion. This bit should be enabled only for 12-bpp B4G4R4A4, 15-bpp B5G5R5A, 16-bpp B5G6R5, 18-bpp B6G6R6 color modes. If enabled the color conversion is performed prior to horizontal scaling. 0 = DISABLE 1 = ENABLE
4	0x0	A_SCAN_COLUMN: Window A Scanning direction 0= Scan in horizontal direction (for 0 or 180 degrees) 1= Scan in vertical direction (for 90 or 270 degrees) 0 = DISABLE 1 = ENABLE
2	X	A_V_DIRECTION: Window A Vertical (Y) drawing Direction 0 = INCREMENT 1 = DECREMENT
0	X	A_H_DIRECTION: Window A Horizontal (X) drawing Direction 0 = INCREMENT 1 = DECREMENT

25.9.44 DC_WIN_A_BYTE_SWAP_0

Window A Byte Swap

Offset: 0x701 | Byte Offset: 0x1c04 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2:0	X	A_BYTE_SWAP: Window A Byte Swap. This controls byte swap of frame data read from memory prior to any data processing in the display module. 00= no byte swap (3 2 1 0) 001= byte swap for each 2-byte word (2 3 0 1) 010= byte swap for each 4-byte word (0 1 2 3) 011= word swap for each 4-byte word (1 0 3 2) 100= byte0 swapped with byte2 (3 0 1 2) 101= left shift every byte (2 1 0 3) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW 4 = SWAP02 5 = SWAPLEFT

25.9.45 DC_WIN_A_COLOR_DEPTH_0

Window A Color Depth For YCbCr data format, Cb and Cr are 8-bit unsigned values. For YUV data format, U and V are 8-bit signed values.

YCbCr422R is similar to YCbCr422P but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P but the U and V are shared vertically.

YCbCr422RA is the same as YCbCr422R in memory, and YUV422RA is the same as YUV422R in memory. However, while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixels are not used (discarded) by the display pipeline. B6x2G6x2R6x2A8 is similar to B8G8R6A8 but with the 2 LSBs zeroed out.

R6x2G6x2B6x2A8 is similar to R8G8B6A8 but with the 2 LSBs zeroed out. Some formats have NVCF_ or T_ prefix aliases for NVidia surface format naming convention, wherein left-to-right component names correspond to MSB to LSB bits within a word. In old names without NVCF_ or T_, left-to-right component names generally map LSB to MSB within a word.

Offset: 0x703 │ Byte Offset: 0x1c0c │ Read/Write: R/W │ Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx001100)

Bit	Reset	Description
5:0	B8G8R8A8	A_COLOR_DEPTH: Window A Color Depth. Supported color depths are: P8 = 8-bpp (palletized) B4G4R4A4 = 12-bpp B4G4R4 B5G5R5A = 15-bpp B5G5R5 AB5G5R5 = 15-bpp B5G5R5 B5G6R5 = 16-bpp B5G6R5 B8G8R8A8 = 32-bpp B8G8R8A8 R8G8B8A8 = 32-bpp R8G8B8A8 YCbCr422 = 16-bpp YCbCr422 packed YUV422 = 16-bpp YUV422 YCbCr420P = 16-bpp YCbCr420 planar YUV420P = 16-bpp YUV420 planar YCbCr422P = 16-bpp YCbCr422 planar YUV422P = 16-bpp YUV422 planar YCbCr422R = 16-bpp YCbCr422 rotated planar YUV422R = 16-bpp YUV422 rotated planar YCbCr422RA= 16-bpp YCbCr422 rotated planar with chroma averaging YUV422RA = 16-bpp YUV422 rotated planar with chroma averaging. Reserved values: 0,1,2,14,15,24,25. 3 = T_P8, P8 4 = T_A4R4G4B4, B4G4R4A4 5 = T_A1R5G5B5, B5G5R5A 6 = T_R5G6B5, B5G6R5 7 = T_R5G5B5A1, AB5G5R5 12 = T_A8R8G8B8, B8G8R8A8 13 = T_A8B8G8R8, R8G8B8A8 16 = T_U8_Y8__V8_Y8, CbYCrY422, YCbCr422 17 = T_U8_Y8__V8_Y8_TRUE, UYVY422, YUV422, TRUE_U8_Y8__V8_Y8 18 = T_Y8___U8___V8_N420, YCbCr420P 19 = T_Y8___U8___V8_N420_TRUE, YUV420P, TRUE_Y8___U8___V8_N420 20 = T_Y8___U8___V8_N422, YCbCr422P 21 = T_Y8___U8___V8_N422_TRUE, YUV422P, TRUE_Y8___U8___V8_N422 22 = T_Y8___U8___V8_N422R, YCbCr422RP, YCbCr422R 23 = T_Y8___U8___V8_N422R_TRUE, YUV422RP, YUV422R, TRUE_Y8___U8___V8_N422R (new formats, reserved for future use) 26 = T_Y8, Y8 27 = T_A4B4G4R4, R4G4B4A4 28 = T_A1B5G5R5, R5G5B5A 29 = T_B5G5R5A1, AR5G5B5 30 = T_X1R5G5B5, B5G5R5X1 31 = T_R5G5B5X1, X1B5G5R5 32 = T_X1B5G5R5, R5G5B5X1 33 = T_B5G5R5X1, X1R5G5B5 34 = T_B5G6R5, R5G6B5 35 = T_B8G8R8A8 36 = T_R8G8B8A8 37 = T_X8R8G8B8, B8G8R8X8 38 = T_X8B8G8R8, R8G8B8X8 39 = T_A8Y8U8V8 40 = T_V8U8Y8A8 41 = T_Y8___U8___V8_N444, YCbCr444P 42 = T_Y8___U8V8_N420, YCrCb420SP 43 = T_Y8___V8U8_N420, YCbCr420SP 44 = T_Y8___U8V8_N422, YCrCb422SP 45 = T_Y8___V8U8_N422, YCbCr422SP 46 = T_Y8___U8V8_N422R, YCrCb422RSP 47 = T_Y8___V8U8_N422R, YCbCr422RSP 48 = T_Y8___U8V8_N444, YCrCb444SP 49 = T_Y8___V8U8_N444, YCbCr444SP 50 = T_A8Y8U8V8_TRUE, TRUE_A8Y8U8V8 51 = T_V8U8Y8A8_TRUE, TRUE_V8U8Y8A8 52 = T_Y8___U8___V8_N444_TRUE, YUV444P, TRUE_Y8___U8___V8_N444 53 = T_Y8___U8V8_N420_TRUE, YVU420SP, TRUE_Y8___U8V8_N420 54 = T_Y8___V8U8_N420_TRUE, YUV420SP, TRUE_Y8___V8U8_N420 55 = T_Y8___U8V8_N422_TRUE, YVU422SP, TRUE_Y8___U8V8_N422 56 = T_Y8___V8U8_N422_TRUE, YUV422SP, TRUE_Y8___V8U8_N422 57 = T_Y8___U8V8_N422R_TRUE, YVU422RSP, TRUE_Y8___U8V8_N422R 58 = T_Y8___V8U8_N422R_TRUE, YUV422RSP, TRUE_Y8___V8U8_N422R 59 = T_Y8___U8V8_N444_TRUE, YUV444SP, TRUE_Y8___U8V8_N444 60 = T_Y8___V8U8_N444_TRUE, YVU444SP TRUE_Y8___V8U8_N444

25.9.46 DC_WIN_A_POSITION_0

Window A Position

This register defines the H position and size of Window A after scaling (if there is any)

Offset: 0x704 │ Byte Offset: 0x1c10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	A_V_POSITION: Window A V Position. This is specified with respect to the top edge of active display area.
12:0	X	A_H_POSITION: Window A H Position. This is specified with respect to the left edge of active display area.

25.9.47 DC_WIN_A_SIZE_0

Window A Size

This register defines the V position and size of Window A after scaling (if there is any)

Note: programming on window size should guarantee the whole window is inside the active area, otherwise extra rows/columns will be fetched and discarded which affects performance.

Offset: 0x705 │ Byte Offset: 0x1c14 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	A_V_SIZE: Window A V Size (lines). This is the vertical size after scaling.
12:0	X	A_H_SIZE: Window A H Size (pixels). This is the horizontal size after scaling.

25.9.48 DC_WIN_A_PRESCALED_SIZE_0

Window A Pre-scaled Size

This register defines Window A pre-scaled size.

The H pre-scaled size is needed to determine how many bytes to fetch from memory per line and this parameter must be programmed exactly as needed taking into account the scaling factor. For planar YUV or YCbCr data formats, this parameter refer to the H pre-scaled of the Y plane.

The total number of lines to be fetched from memory is determined by post-scale V size but V pre-scaled size is needed to 'clamp' the last valid line if the vertical DDA is exactly or slightly beyond the specified V pre-scaled size.

H pre-scaled size ideally should be in terms of pixels but then hardware needs to convert this precisely to bytes to determine the amount of data to request from memory.

Offset: 0x706 │ Byte Offset: 0x1c18 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	A_V_PRESCALED_SIZE: Window A V Pre-scaled Size (lines). In 420P/422R/422RA formats, it must be even.
14:0	X	A_H_PRESCALED_SIZE: Window A H Pre-scaled Size (bytes). In 420P and 422P formats, it must be even.

25.9.49 DC_WIN_A_H_INITIAL_DDA_0

Window A H Initial DDA

Design Note: the first pixel of pre-scaled image is always used to output the first pixel so essentially this is the same as forcing the H Initial DDA integer portion to 1 initially even though the user typically programs this to 0. If it makes the implementation easier, it is possible to force software to program the Initial DDA integer portion to 1. Similarly, with the V Initial DDA.

Offset: 0x707 │ Byte Offset: 0x1c1c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	A_H_INITIAL_DDA: Window A H Initial DDA (4.12). This is typically programmed to 0.0

25.9.50 DC_WIN_A_V_INITIAL_DDA_0

Window A V Initial DDA

Offset: 0x708 │ Byte Offset: 0x1c20 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	A_V_INITIAL_DDA: Window A V Initial DDA (4.12). This is typically programmed to 0.0 for both non-interlaced and interlaced sources.

25.9.51 DC_WIN_A_DDA_INCREMENT_0

Window A DDA Increment

DDA increment is typically calculated by dividing (pre-scaled size in pixels - 1) by (post-scaled size in pixels - 1). The result should be rounded up and expressed as 4.12 format (4-bit integer and 12-bit fraction). For non-filtered image this value can be slightly larger so that it is not missing the last row/column. Reference programming values (H and V should be calculated separately depending on filter on/off and sizes):

- Filter on: min(round((prescaled_size_in_pixels - 1) * 0x1000 / (post_scaled_size_in_pixels - 1)), MAX)
- Filter off: min(round(prescaled_size_in_pixels * 0x1000 / (post_scaled_size_in_pixels - 1) - 0.5), MAX)

Where the value of MAX is as follows:

For V_DDA_INCREMENT: 15.0 (0xF000)

For H_DDA_INCREMETN: 4.0 (0x4000) for 4 Bytes/pix formats.

8.0 (0x8000) for 2 Bytes/pix formats.

They are theoretically the biggest values that guarantee not displaying beyond an image boundary. If the DDA increment is less than 1.0, then the image is upscaled. If the DDA increment is more than 1.0, then the image is downscaled.

Offset: 0x709 │ Byte Offset: 0x1c24 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	A_V_DDA_INCREMENT: Window A Vertical DDA Increment (4.12). This should be set to 1.0 if there is no scaling. Maximum value is 15.0 regardless of the number of bytes per pixel.

Bit	Reset	Description
15:0	X	A_H_DDA_INCREMENT: Window A Horizontal DDA Increment (4.12). This should be set to 1.0 if there is no scaling. The maximum value for downscaling depends on the number of bytes per pixel. For 4-byte/pixel modes (32-bpp) the maximum value is 4.0 and for all other modes the maximum value is 8.0.

25.9.52 DC_WIN_A_LINE_STRIDE_0

Window A Line Stride

Offset: 0x70a │ Byte Offset: 0x1c28 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	A_UV_LINE_STRIDE: Window A Line Stride for Chroma. This is stride (in bytes) for planar YUV or YCbCr data formats for the chroma plane, with the restriction that it must be programmed to be multiples of 4 (16 if tiled or in horizontal flipping) This is not used (ignored) for other non-planar data formats.
15:0	X	A_LINE_STRIDE: Window A Line Stride. This is stride (in bytes) for all non-planar data formats. If the memory surface is tiled, the stride needs to be a multiple of 16. If H_DIRECTION of window A is set to DECREMENT, the stride also needs to be a multiple of 16. For planar YUV or YCbCr data formats, this is stride (in bytes) for the luma plane with the restriction that it must be multiples of 8 (16 if tiled or in horizontal flipping) For tiled surface this value may affect starting address of a window. Refer to the comment of START_ADDR for detail.

25.9.53 DC_WIN_A_BUFFER_ADDR_MODE_0

Memory Controller Tiling definitions

Offset: 0x70d │ Byte Offset: 0x1c34 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxxxxxxxxxx0)

Bit	Reset	Description
16	0x0	A_UV_TILE_MODE: Window A Memory surface tiling mode. For YUV planar pixel format, this specifies tiling mode for the U/V plane. 0 = LINEAR 1 = TILED
0	0x0	A_TILE_MODE: Window A Memory surface tiling mode. For YUV planar pixel format, this specifies tiling mode for the Y plane. 0 = LINEAR 1 = TILED

25.9.54 DC_WIN_A_DV_CONTROL_0

Window A Digital Vibrance Control

If enabled, Digital Vibrance is applied after H and V scaling and after color palette or color space conversion logic but before color keying multiplexer and before cursor multiplexer.

- After DV, new R = R + (2R - G - B) * FR, where FR is fraction from 0 to 7/8
- After DV, new G = G + (2G - R - B) * FG, where FG is fraction from 0 to 7/8
- After DV, new B = B + (2B - R - G) * FB, where FB is fraction from 0 to 7/8

Offset: 0x70e │ Byte Offset: 0x1c38 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18:16	X	A_DV_CONTROL_B: Digital Vibrance control for B
10:8	X	A_DV_CONTROL_G: Digital Vibrance control for G
2:0	X	A_DV_CONTROL_R: Digital Vibrance control for R

25.9.55 DC_WIN_A_BLEND_NOKEY_0

Blend Control for this window area where color key is enabled but the pixel color is not within the color key range (color key not match). This is valid for all overlapping condition but only if there is no overlap with other window with higher priority color key enabled and color key not match.

Class: Display Color Keying and Blending

Color keying and blending of the display windows are done prior to cursor blending.

Cursor always goes on top of the blended windows. Blending is controlled independently on each possible overlap area of the display windows. If 3 windows are enabled there are 7 possible overlap area combinations. For every window in each overlap area combination, color key can be disabled or enabled. Also, for every window in each overlap area combination, there is a corresponding window blend control parameter and a window blend weight parameter. The window blend control parameter is always effective but the window blend weight is not always used. The window blend weight can also be derived from pixel alpha value or from the reverse of other overlapping windows weight.

Color keying has the highest priority for display window blending. Color key consists of a range of color which is searched independently for each window. If more than 1 windows color key is enabled, then Window A color key has the highest priority, followed by Window B color key, and then followed by Window C color key. Two sets of color key ranges (Color Key 0 and Color Key 1) can be defined; they are shared for all windows. It is possible to use both color key sets for the same window or for two separate windows.

The two sets of color key ranges should not overlap. If they do, the overlapped colors are treated as if they are part of Color Key 0 and not part of Color Key 1.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for any overlap condition and the window pixel is not within the color key range (key not match), then the window pixel will not be blended with other overlapping window pixels but it will be weighted. The weight is not dependent on the overlap condition it is controlled by the same set of parameters for all overlapping condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for a particular overlap condition and the window pixel is within the color key range (key match), then the window pixel will be weighted and blended with other overlapping pixels and this is controlled separately for each overlap condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is disabled then the window pixel will be blended with other overlapping pixels and this is controlled separately for each overlap condition.

Display Color Key parameters

For B4G4R4A4, B5G6R5A, or B5G6R5 mode, the color key should be compared prior to color conversion to 24-bpp and unused least significant bits of the pixel are filled with zeros.

For palletized mode, the color key is compared prior to the color palette, and the palletized color is compared against the green color key values/mask.

For YUV mode, U and V are offset by +128 before performing the color key comparison. In all cases, the color key is compared prior to the horizontal/vertical scaling filter and prior to digital vibrance control.

Both upper and lower values are inclusive.

Offset: 0x70f │ Byte Offset: 0x1c3c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	A_BLEND_WEIGHT1_NOKEY: Window blend weight 1 for color key not match areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	A_BLEND_WEIGHT0_NOKEY: Window blend weight 0 for color key not match areas. For alpha weight, this is used for 1-bit alpha with value of 0.
0	X	A_BLEND_CONTROL_NOKEY: Window blend control for color key not match areas. 0 = Fix weight using window blend weight 0 for color key not matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT

25.9.56 DC_WIN_A_BLEND_1WIN_0

Blend Control for this window area where it does not overlap with other windows.

Offset: 0x710 │ Byte Offset: 0x1c40 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	A_BLEND_WEIGHT1_1WIN: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	A_BLEND_WEIGHT0_1WIN: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
2	X	A_BLEND_CONTROL_1WIN: Window blend control in area where it does not overlap with other windows and either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT
1:0	X	A_CKEY_ENABLE_1WIN: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

25.9.57 DC_WIN_A_BLEND_2WIN_B_0

Blend Control for this window area that overlaps with window B only.

Offset: 0x711 | Byte Offset: 0x1c44 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	A_BLEND_WEIGHT1_2WIN_B: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	A_BLEND_WEIGHT0_2WIN_B: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	A_BLEND_CONTROL_2WIN_B: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 3 = Premultipled weight; in this case, the window blend weight is 1. But other windows that use DEPENDENT_WEIGHT get (1 - alpha value), like ALPHA_WEIGHT. This is only valid if the window color format includes an alpha value. This allows a blending equation like: result = win_a * 1 + (1 - win_a_alpha) * win_c. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT 3 = PREMULT_WEIGHT
1:0	X	A_CKEY_ENABLE_2WIN_B: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

25.9.58 DC_WIN_A_BLEND_2WIN_C_0

Blend Control for this window area that overlaps with window C only.

Offset: 0x712 | Byte Offset: 0x1c48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	A_BLEND_WEIGHT1_2WIN_C: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	A_BLEND_WEIGHT0_2WIN_C: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.

Bit	Reset	Description
3:2	X	A_BLEND_CONTROL_2WIN_C: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. Only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 3 = Premultipled weight; in this case, the window blend weight is 1. But other windows that use DEPENDENT_WEIGHT get (1 - alpha value), like ALPHA_WEIGHT. This is only valid if the window color format includes an alpha value. This allows a blending equation like: result = win_a * 1 + (1 - win_a_alpha) * win_b. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT 3 = PREMULT_WEIGHT
1:0	X	A_CKEY_ENABLE_2WIN_C: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

25.9.59 DC_WIN_A_BLEND_3WIN_BC_0

Blend Control for this window area that overlaps with windows B and C only.

Offset: 0x713 │ Byte Offset: 0x1c4c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	A_BLEND_WEIGHT1_3WIN_BC: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	A_BLEND_WEIGHT0_3WIN_BC: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.

Bit	Reset	Description
3:2	X	A_BLEND_CONTROL_3WIN_BC: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 3 = Premultipled weight; in this case, the window blend weight is 1. But other windows that use DEPENDENT_WEIGHT get (1 - alpha value), like ALPHA_WEIGHT. This is only valid if the window color format includes an alpha value 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT 3 = PREMULT_WEIGHT
1:0	X	A_CKEY_ENABLE_3WIN_BC: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

25.9.60 DC_WIN_A_GLOBAL_ALPHA_0

Window A Global Alpha

When enabled, per pixel alpha values are scaled by GLOBAL_ALPHA before blending: a' = a * GLOBAL_ALPHA/255. When per pixel alpha is not used in blending, global alpha has no affect.

Note: 1- and 4-bits per pixel alpha are quantized back to 1 and 4 bits, respectively, after scaling. For the 1-bit alpha case, finer scaling can be achieved by changing BLEND_WEIGHT* instead.

Offset: 0x715 │ Byte Offset: 0x1c54 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxxxxxxxxxx)

Bit	Reset	Description
16	DISABLE	A_GLOBAL_ALPHA_ENABLE: When enabled, color palette *must* also be enabled and programmed. 0 = DISABLE 1 = ENABLE
7:0	X	A_GLOBAL_ALPHA: Global alpha: 0..255 map to 0.0 – 1.0. i.e., 255 yields no-operation. For power, better to set GLOBAL_ALPHA_ENABLE=DISABLE when GLOBAL_ALPHA=255.

25.10 WINBUF_A Registers

The registers under DC_WINBUF are triple-buffered.

25.10.1 DC_WINBUF_A_START_ADDR_0

Window A Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally, START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specifies the address offsets of the pixel at the beginning of the window with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see ”For linear address mode” in Programming Restrictions below.)

Note that "beginning of the window" here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting Address Calculation

These formulae are same for both tiled and linear modes. However, for linear mode, the addresses calculated are real physical addresses. For tiled mode, these addresses will be translated to the real physical addresses by the memory controller client before being used.

- When a window is host triggered, starting address of a window is calculated as follows by hardware:
  - non-YUV-planar modes:

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar modes:

    Y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.
- When a window is non-host triggered, starting address of a window buffer is calculated as below.
  - non-YUV-planar mode:

    starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar mode:

    Y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

    buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

i. For tiled address mode:

Image surface can only aligned to multiples of 256, thus the following restrictions.

- START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.
- BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256
- LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16
- ADDR_H_OFFSET needs to be even in YUV planar format, or a multiple of bytes per pixel in other formats. If H_DIRECTION=DECREMENT, however, it should point to last valid byte, which is an odd offset.
- ADDR_V_OFFSET needs to be a multiple of 2 in YUV planar format, unless H_DIRECTION=DECREMENT, but with no restrictions on other color formats.

ii. For linear address mode:

Image surface can be aligned 2, 4 or 8 bytes, depending on the color formats.

As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formulae in Starting Address Calculation above still hold)

- For all formats:
  - START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.
- For 16-bpp formats,
  - (START_ADDR+H_OFFSET) need to be a multiple of 2.
- For 32-bpp formats,
  - (START_ADDR+H_OFFSTE) needs to be a multiple of 4.
- For YUV planar formats:
  - BUF_STRIDE, UV_BUF_STRIDE:

    BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

    or as a stricter constraint: BUF_STRIDE is a multiple of 8, UV_BUF_STRIDE is a multiple of 4.
  - LINE_STRIDE, UV_LINE_STRIDE:

    LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

    LINE_STRIDE needs to be a multiple of 8, UV_LINE_STRIDE needs to be a multiple of 4.
  - ADDR_H_OFFSET: Needs to be even unless H_DIRECTION=DECREMENT, in which case should point to the last byte pixel, which is at an odd offset. .
  - ADDR_V_OFFSET: Needs to be even unless V_DIRECTION=DECREMENT, in which case should point to the last valid line, which is at an odd offset. .

iii. Memory allocation for non-host triggered case:

When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because display will use the same luma or chroma line strides for all lines in the frame. Also buffer wraparound must not occur in the middle of the displayed part of the frame.

The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not

switched until frame end flag is active. In the case where one buffer corresponds to one frame then frame start and frame end flag are active every time a buffer index is sent.

Offset: 0x800 | Byte Offset: 0x2000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR: Window A Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

25.10.2 DC_WINBUF_A_START_ADDR_NS_0

Window A Shadowed Start Address

Offset: 0x801 | Byte Offset: 0x2004 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_NS: Window A Shadowed Start Address. This is ARM set shadow of Start Address.

25.10.3 DC_WINBUF_A_START_ADDR_U_0

Window A Start Address for U plane

Offset: 0x802 | Byte Offset: 0x2008 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_U: Window A Start Address for U plane. This is a byte address.

25.10.4 DC_WINBUF_A_START_ADDR_U_NS_0

Window A Shadowed Start Address for U plane

Offset: 0x803 | Byte Offset: 0x200c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_U_NS: Window A Shadowed Start Address for U plane. This is ARM set shadow register of U start address

25.10.5 DC_WINBUF_A_START_ADDR_V_0

Window A Start Address for V plane

Offset: 0x804 | Byte Offset: 0x2010 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_V: Window A Start Address for V plane. This is a byte address.

25.10.6 DC_WINBUF_A_START_ADDR_V_NS_0

Window A Shadowed Start Address for V plane

Offset: 0x805 | Byte Offset: 0x2014 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_START_ADDR_V_NS: Window A Shadowed Start Address for V plane. This is ARM set shadow register of U start address

25.10.7 DC_WINBUF_A_ADDR_H_OFFSET_0

Window A Horizontal Address Offset

Offset: 0x806 | Byte Offset: 0x2018 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_ADDR_H_OFFSET: Window A Horizontal address offset. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by hardware.

25.10.8 DC_WINBUF_A_ADDR_H_OFFSET_NS_0

Window A Shadowed Horizontal Address Offset

Offset: 0x807 | Byte Offset: 0x201c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_ADDR_H_OFFSET_NS: Window A Shadowed Horizontal address offset. This is ARM set shadow of ADDR_H_OFFSET

25.10.9 DC_WINBUF_A_ADDR_V_OFFSET_0

Window A Vertical Address Offset

Offset: 0x808 | Byte Offset: 0x2020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_ADDR_V_OFFSET: Window A Vertical address offset. This is a line number. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies vertical offset of Y plane. The vertical offsets of U/V plane is derived by hardware.

25.10.10 DC_WINBUF_A_ADDR_V_OFFSET_NS_0

Window A Shadowed Vertical Address Offset

Offset: 0x809 | Byte Offset: 0x2024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_ADDR_V_OFFSET_NS: Window A Shadowed Vertical address offset. This is the ARM set shadow of ADDR_V_OFFSET.

25.10.11 DC_WINBUF_A_UFLOW_STATUS_0

Window A FIFO Underflow Status Register

Offset: 0x80a │ Byte Offset: 0x2028 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
23:0	X	UFLOW_COUNT: Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.

25.10.12 DC_WINBUF_A_SURFACE_WEIGHT_0

Window A Surface Weights

Offset: 0x80c │ Byte Offset: 0x2030 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
6:5	X	A_SURFACE_WEIGHT_V: Window A V Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
4:3	X	A_SURFACE_WEIGHT_U: Window A U or UV Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
2:1	X	A_SURFACE_WEIGHT_Y: Window A Y or packed Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
0	0x0	A_SURFACE_WEIGHT_OVERRIDE: Weight Override 0 = DISABLE 1 = ENABLE

25.10.13 DC_WINBUF_A_START_ADDR_HI_0

Window A Higher 32 bits of Start Address

Makes the start address 64 bits by adding the higher 32 bits below.

Offset: 0x80d │ Byte Offset: 0x2034 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	A_START_ADDR_HI: Window A Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

25.10.14 DC_WINBUF_A_START_ADDR_HI_NS_0

Window A Shadowed Higher 32 bits of Start Address

Offset: 0x80e │ Byte Offset: 0x2038 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	A_START_ADDR_HI_NS: Window A Shadowed Start Address. This is ARM set shadow of Start Address.

25.10.15 DC_WINBUF_A_START_ADDR_HI_U_0

Window A Higher 32 bits of Start Address for U Plane

Offset: 0x80f │ Byte Offset: 0x203c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	A_START_ADDR_HI_U: Window A Start Address for U plane. This is a byte address.

25.10.16 DC_WINBUF_A_START_ADDR_HI_U_NS_0

Window A Shadowed Higher 32 bits of Start Address for U Plane

Offset: 0x810 │ Byte Offset: 0x2040 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	A_START_ADDR_HI_U_NS: Window A Shadowed Higher 32 bits of Start Address for U plane. This is ARM set shadow register of U start address

25.10.17 DC_WINBUF_A_START_ADDR_HI_V_0

Window A Higher 32 bits of Start Address for V Plane

Offset: 0x811 │ Byte Offset: 0x2044 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	A_START_ADDR_HI_V: Window A Higher 32 bits of Start Address for V plane. This is a byte address.

25.10.18 DC_WINBUF_A_START_ADDR_HI_V_NS_0

Window A Shadowed Higher 32 bits of Start Address for V Plane

Offset: 0x812 | Byte Offset: 0x2048 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	A_START_ADDR_HI_V_NS: Window A Shadowed Higher 32 bits of Start Address for V plane. This is ARM set shadow register of U start address

25.10.19 DC_WINBUF_A_UFLOW_CTRL_0

When the underflow debug mode is enabled. Instead of sending out the last valid pixel, DBG_PIXEL is sent out.

Offset: 0x824 | Byte Offset: 0x2090 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	A_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

25.10.20 DC_WINBUF_A_UFLOW_DBG_PIXEL_0

Offset: 0x825 | Byte Offset: 0x2094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	A_UFLOW_DBG_PIXEL

25.10.21 DC_WINBUF_A_UFLOW_THRESHOLD_0

Offset: 0x826 | Byte Offset: 0x2098 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	A_UFLOW_THRESHOLD

25.10.22 DC_WINBUF_A_SPOOL_UP_0

Spool up time configuration for different windows related logic.

SPOOL_UP_CTRL = MAX - This is the current Tegra 4 behavior. This allows for memfetch to start fetching for a window (irrespective of its relative position on the screen), at the very beginning of the vblank of the next frame. Since the spool up time is maximum, this should have the least probability of underflow of the display pipe. However, on the down side, since all three windows would start fetching at the same time, and since display is a high priority client for MC, it has a high potential to starve out other clients.

SPOOL_UP_CTRL = PROGRAMMABLE, SPOOL_UP_DURATION = 1 - This corresponds to the Tegra 3 behavior. Since the fetch starts only 1 line prior to the active scan line of a window, this has high probability of causing underflow, but has a lesser impact on the other MC clients.

So, this register programmability programs a method to modulate the spool up time between the best and worst, if required, when other MC clients are present. For example when GPU/VIC is active, we may not want a very aggressive spool up and program some definite spool up duration. So, spool up time is a general knob, which would be a function of the resolution, bpp, active clients, memory speed, etc.

SPOOL_UP_EDGE :- By default the memfetch start signal generated by the hvtgen logic is one line clock wide.

For SPOOL_UP_CTRL = MAX:

- In this case if SPOOL_UP_EDGE is programmed to NEGEDGE. This ensures that if any act_req is sent during the time vcounter = 0(at the beginning of the frame). The activate value is loaded into the active state and the memfetch would see the latest updated value. The downside of programming it to NEGDGE is that 1 line clock worth of spool up is lost. This is required based upon Tegra compatibility and the way tests are written.
- In this case if SPOOL_UP_EDGE is programmed to POSEDGE. Then the memfetch would start fetching at the very beginning of the frame start pulse (1 line clock wide) and would give maximum spool up time. However, this may give test failure and potential Tegra compatibility issues as described above.

FOR SPOOL_UP_CTRL = PROGRAMMABLE :

- Special care must be taken here. If SPOOL_UP_DURATION is set to 1. SPOOL_UP_EDGE must be set to POSEDGE to allow for at least 1 line worth of spool up.
- If SPOOL_UP_DURATION is > 1. Then SPOOL_UP_EDGE can be either POSEDGE or NEGEDGE. If it is NEGEDGE, then the effective spool up duration would be 1 line clock less than the SPOOL_UP_DURATION.

Offset: 0x827 │ Byte Offset: 0x209c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
28:16	X	A_SPOOL_UP_DURATION
1	0x0	A_SPOOL_UP_EDGE: 0 = NEGEDGE 1 = POSEDGE
0	0x0	A_SPOOL_UP_CTRL: 0 = MAX 1 = PROGRAMMABLE

25.10.23 DC_WINBUF_A_SCALEFACTOR_THRESHOLD_0

Registers for latency allowance scaling. Threshold is in terms of 64 byte atoms. Display memfetch compares its buffer occupancy with the threshold and generates indication to MC if its buffer occupancy is above HWM or below LWM.

Offset: 0x828 │ Byte Offset: 0x20a0 │ Read/Write: R/W │ Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:16	0xffff	A_SF_LWM_THRESHOLD
15:0	0xffff	A_SF_HWM_THRESHOLD

25.10.24 DC_WINBUF_A_LATENCY_THRESHOLD_0

Register for ready for latency event like DVFS. Threshold is in terms of 64 byte atoms. Display memfetch compares its buffer occupancy with the threshold and generates indication to MC if its buffer occupancy is above the threshold.

Offset: 0x829 │ Byte Offset: 0x20a4 │ Read/Write: R/W │ Reset: 0x0000ffff (0bxxxxxxxxxxxxxxxx1111111111111111)

Bit	Reset	Description
15:0	0xffff	A_RDY4LATENCY_THRESHOLD

25.10.25 DC_WINBUF_A_MEMFETCH_DEBUG_STATUS_0

Memfetch Debug Status

Offset: 0x82a │ Byte Offset: 0x20a8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15	0x0	B_ALIGN_FIFO_NON_IDLE: align FIFO idle at end of frame. 0= align FIFO is idle at end of frame. 1= align FIFO is not idle at end of frame
14	0x0	B_PIPE1_FIFO_NON_IDLE: pipe1 FIFO idle at end of frame. 0= pipe1 FIFO is idle at end of frame. 1= pipe1 FIFO is not idle at end of frame
13	0x0	B_PIPE0_FIFO_NON_IDLE: pipe0 FIFO idle at end of frame. 0= pipe0 FIFO is idle at end of frame. 1= pipe0 FIFO is not idle at end of frame
12	0x0	B_FRAME_FIFO_NON_IDLE: frame FIFO idle at end of frame. 0= frame FIFO is idle at end of frame. 1= frame FIFO is not idle at end of frame
11	0x0	B_SURFACE_FIFO_NON_IDLE: surface FIFO idle at end of frame. 0= surface FIFO is idle at end of frame. 1= surface FIFO is not idle at end of frame
10	0x0	B_TAG_FIFO_NON_IDLE: tag FIFO idle at end of frame. 0= tag FIFO is idle at end of frame. 1= tag FIFO is not idle at end of frame
9	0x0	B_RSP_FIFO_NON_IDLE: Response FIFO idle at end of frame. 0= response FIFO is idle at end of frame. 1= response FIFO is not idle at end of frame
8	0x0	B_REQ_FIFO_NON_IDLE: Request FIFO idle at end of frame. 0= request FIFO is idle at end of frame. 1= request FIFO is not idle at end of frame
7	0x0	B_DP_VALID_NON_IDLE: DP lines valid at end of frame. 0= DP lines are not valid at end of frame. 1= DP lines are valid at end of frame
6	0x0	B_V_SSM_NON_IDLE: V SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
5	0x0	B_U_SSM_NON_IDLE: U SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
4	0x0	B_Y_SSM_NON_IDLE: Y SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
3	0x0	B_DP_POOL_AVAIL: DP thinks there is data in the pool at end of frame. 0= No pool data available at end of frame. 1= pool data available at end of frame
2	0x0	B_POOL_NOT_EMPTY: Status of pool at end of frame. 0= Pool is empty at end of frame. 1= Pool is not empty at end of frame

Bit	Reset	Description
1	0x0	B_UNDERFLOW_LINE1: Underflow of line0 0= No underflow 1= line0 underflowed
0	0x0	B_UNDERFLOW_LINE0: Underflow of line0 0= No underflow 1= line0 underflowed

25.10.26 DC_WINBUF_A_MEMFETCH_CONTROL_0

Memfetch Control Register

Offset: 0x82b │ Byte Offset: 0x20ac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	A_MEMFETCH_CLK_GATE_OVERRIDE: Disable clock-gating of memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE
0	0x0	A_MEMFETCH_RESET: Reset memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE

25.10.27 DC_WINBUF_A_OCCUPANCY_THROTTLE_0

Offset: 0x82c │ Byte Offset: 0x20b0 │ Read/Write: R/W │ Reset: 0xffff0000 (0b1111111111111111xxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:16	0xffff	A_OCCUPANCY_MAX_THRESHOLD
0	0x0	A_OCCUPANCY_THROTTLE_MODE: 0 = DISABLE 1 = ENABLE

25.10.28 DC_WINBUF_A_SCRATCH_REGISTER_0_0

Offset: 0x82d │ Byte Offset: 0x20b4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	A_SCRATCH_REGISTER_0:Scratch register 0

25.10.29 DC_WINBUF_A_SCRATCH_REGISTER_1_0

Offset: 0x82e │ Byte Offset: 0x20b8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	A_SCRATCH_REGISTER_1:Scratch register 1

25.11 Window B (WINC_B) Registers

These registers control window B parameters

25.11.1 DC_WINC_B_COLOR_PALETTE_0

Window B Color Palette

This is used for palletized data format (color depth of 8-bpp or less) or for gamma correction for non-palletized data formats (color depth of more than 8-bpp).

Each window has its own color palette which consists of three 256x8 register file which can be written by host and indexed (read) by the window.

For palletized data format less than 8 bpp, the pixel data is aligned to least significant bits of the palette index (address) and the remaining upper bits are filled with the corresponding bits of the Palette Color Extension. For example, for 4-bpp mode, the pixel data occupies bits 3-0 of the palette index, and bits 7-4 of the palette index are set to bits 7-4 of the Palette Color Extension.

Note that host read is assumed to be not needed - software can cache the color palette in system memory.

This is an array of 256 identical register entries; the register fields below apply to each entry.

Offset: 0x500..0x5ff │ Byte Offset: 0x1400..0x17fc │ Read/Write: WO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	B_COLOR_PALETTE_B: Blue Color Palette
15:8	X	B_COLOR_PALETTE_G: Green Color Palette
7:0	X	B_COLOR_PALETTE_R: Red Color Palette

25.11.2 DC_WINC_B_PALETTE_COLOR_EXT_0

Window B Palette Color Extension

Palette extension for 1-bpp, 2-bpp, and 4-bpp. These bits provide the upper most significant bits for indexing the color palette.

Offset: 0x600 │ Byte Offset: 0x1800 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:1	X	B_PALETTE_COLOR_EXT: Window B Palette Color Extension bits 7-1 are used for 1-bpp mode bits 7-2 are used for 2-bpp mode bits 7-4 are used for 4-bpp mode

25.11.3 DC_WINC_B_H_FILTER_P00_0

Window B Horizontal Filter phase 00

Horizontal Scaling Filter Coefficients

Horizontal scaling filter is a 6-tap filter with 4-bit positional phase.

- Coefficients 0 and 5 are 3-bit signed values ranging from -4 to 3.
- Coefficients 1 and 4 are 5-bit signed values ranging from -16 to 15.
- Coefficients 2 and 3 are 8-bit unsigned values ranging from 0 to 128.

- Coefficient 0 is the multiplier for the earliest pixel (P0) in the group of 6 pixels and coefficient 5 is the multiplier for the latest pixel (P5) in the group. The output pixel positional phase is defined as centered in P2 if the positional phase is 0 or proportionally in between P2 and P3 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically, and software should never program the sum of all coefficients for a phase to be more than 128. For each horizontal positional phase, the 6 filter coefficients requires 32 register bits. Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Offset: 0x601 │ Byte Offset: 0x1804 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P00C5: Phase 00 coefficient 5 (typically 0)
28:24	X	B_H_FILTER_P00C4: Phase 00 coefficient 4 (typically 0)
23:16	X	B_H_FILTER_P00C3: Phase 00 coefficient 3 (typically 0)
15:8	X	B_H_FILTER_P00C2: Phase 00 coefficient 2 (typically 128)
7:3	X	B_H_FILTER_P00C1: Phase 00 coefficient 1 (typically 0)
2:0	X	B_H_FILTER_P00C0: Phase 00 coefficient 0 (typically 0)

25.11.4 DC_WINC_B_H_FILTER_P01_0

Window B Horizontal Filter phase 01

Offset: 0x602 │ Byte Offset: 0x1808 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P01C5: Phase 01 coefficient 5 (typically 1)
28:24	X	B_H_FILTER_P01C4: Phase 01 coefficient 4 (typically -2)
23:16	X	B_H_FILTER_P01C3: Phase 01 coefficient 3 (typically 8)
15:8	X	B_H_FILTER_P01C2: Phase 01 coefficient 2 (typically 124)
7:3	X	B_H_FILTER_P01C1: Phase 01 coefficient 1 (typically -4)
2:0	X	B_H_FILTER_P01C0: Phase 01 coefficient 0 (typically 1)

25.11.5 DC_WINC_B_H_FILTER_P02_0

Window B Horizontal Filter phase 02

Offset: 0x603 │ Byte Offset: 0x180c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P02C5: Phase 02 coefficient 5 (typically 1)
28:24	X	B_H_FILTER_P02C4: Phase 02 coefficient 4 (typically -5)
23:16	X	B_H_FILTER_P02C3: Phase 02 coefficient 3 (typically 17)
15:8	X	B_H_FILTER_P02C2: Phase 02 coefficient 2 (typically 122)

Bit	Reset	Description
7:3	X	B_H_FILTER_P02C1: Phase 02 coefficient 1 (typically -8)
2:0	X	B_H_FILTER_P02C0: Phase 02 coefficient 0 (typically 1)

25.11.6 DC_WINC_B_H_FILTER_P03_0

Window B Horizontal Filter phase 03

Offset: 0x604 | Byte Offset: 0x1810 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P03C5: Phase 03 coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P03C4: Phase 03 coefficient 4 (typically -7)
23:16	X	B_H_FILTER_P03C3: Phase 03 coefficient 3 (typically 27)
15:8	X	B_H_FILTER_P03C2: Phase 03 coefficient 2 (typically 115)
7:3	X	B_H_FILTER_P03C1: Phase 03 coefficient 1 (typically -11)
2:0	X	B_H_FILTER_P03C0: Phase 03 coefficient 0 (typically 2)

25.11.7 DC_WINC_B_H_FILTER_P04_0

Window B Horizontal Filter phase 04

Offset: 0x605 | Byte Offset: 0x1814 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P04C5: Phase 04 coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P04C4: Phase 04 coefficient 4 (typically -9)
23:16	X	B_H_FILTER_P04C3: Phase 04 coefficient 3 (typically 37)
15:8	X	B_H_FILTER_P04C2: Phase 04 coefficient 2 (typically 109)
7:3	X	B_H_FILTER_P04C1: Phase 04 coefficient 1 (typically -13)
2:0	X	B_H_FILTER_P04C0: Phase 04 coefficient 0 (typically 2)

25.11.8 DC_WINC_B_H_FILTER_P05_0

Window B Horizontal Filter phase 05

Offset: 0x606 | Byte Offset: 0x1818 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P05C5: Phase 05 coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P05C4: Phase 05 coefficient 4 (typically -11)
23:16	X	B_H_FILTER_P05C3: Phase 05 coefficient 3 (typically 47)

Bit	Reset	Description
15:8	X	B_H_FILTER_P05C2: Phase 05 coefficient 2 (typically 102)
7:3	X	B_H_FILTER_P05C1: Phase 05 coefficient 1 (typically -15)
2:0	X	B_H_FILTER_P05C0: Phase 05 coefficient 0 (typically 3)

25.11.9 DC_WINC_B_H_FILTER_P06_0

Window B Horizontal Filter phase 06

Offset: 0x607 | Byte Offset: 0x181c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P06C5: Phase 06 coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P06C4: Phase 06 coefficient 4 (typically -13)
23:16	X	B_H_FILTER_P06C3: Phase 06 coefficient 3 (typically 56)
15:8	X	B_H_FILTER_P06C2: Phase 06 coefficient 2 (typically 94)
7:3	X	B_H_FILTER_P06C1: Phase 06 coefficient 1 (typically -15)
2:0	X	B_H_FILTER_P06C0: Phase 06 coefficient 0 (typically 3)

25.11.10 DC_WINC_B_H_FILTER_P07_0

Window B Horizontal Filter phase 07

Offset: 0x608 | Byte Offset: 0x1820 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P07C5: Phase 07 coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P07C4: Phase 07 coefficient 4 (typically -14)
23:16	X	B_H_FILTER_P07C3: Phase 07 coefficient 3 (typically 67)
15:8	X	B_H_FILTER_P07C2: Phase 07 coefficient 2 (typically 85)
7:3	X	B_H_FILTER_P07C1: Phase 07 coefficient 1 (typically -16)
2:0	X	B_H_FILTER_P07C0: Phase 07 coefficient 0 (typically 3)

25.11.11 DC_WINC_B_H_FILTER_P08_0

Window B Horizontal Filter phase 08

Offset: 0x609 | Byte Offset: 0x1824 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P08C5: Phase 08 coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P08C4: Phase 08 coefficient 4 (typically -15)

Bit	Reset	Description
23:16	X	B_H_FILTER_P08C3: Phase 08 coefficient 3 (typically 76)
15:8	X	B_H_FILTER_P08C2: Phase 08 coefficient 2 (typically 76)
7:3	X	B_H_FILTER_P08C1: Phase 08 coefficient 1 (typically -15)
2:0	X	B_H_FILTER_P08C0: Phase 08 coefficient 0 (typically 3)

25.11.12 DC_WINC_B_H_FILTER_P09_0

Window B Horizontal Filter phase 09

Offset: 0x60a │ Byte Offset: 0x1828 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P09C5: Phase 09 coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P09C4: Phase 09 coefficient 4 (typically -16)
23:16	X	B_H_FILTER_P09C3: Phase 09 coefficient 3 (typically 85)
15:8	X	B_H_FILTER_P09C2: Phase 09 coefficient 2 (typically 67)
7:3	X	B_H_FILTER_P09C1: Phase 09 coefficient 1 (typically -14)
2:0	X	B_H_FILTER_P09C0: Phase 09 coefficient 0 (typically 3)

25.11.13 DC_WINC_B_H_FILTER_P0A_0

Window B Horizontal Filter phase 0A

Offset: 0x60b │ Byte Offset: 0x182c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P0AC5: Phase 0A coefficient 5 (typically 3)
28:24	X	B_H_FILTER_P0AC4: Phase 0A coefficient 4 (typically -15)
23:16	X	B_H_FILTER_P0AC3: Phase 0A coefficient 3 (typically 94)
15:8	X	B_H_FILTER_P0AC2: Phase 0A coefficient 2 (typically 56)
7:3	X	B_H_FILTER_P0AC1: Phase 0A coefficient 1 (typically -13)
2:0	X	B_H_FILTER_P0AC0: Phase 0A coefficient 0 (typically 3)

25.11.14 DC_WINC_B_H_FILTER_P0B_0

Window B Horizontal Filter phase 0B

Offset: 0x60c │ Byte Offset: 0x1830 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P0BC5: Phase 0B coefficient 5 (typically 3)

Bit	Reset	Description
28:24	X	B_H_FILTER_P0BC4: Phase 0B coefficient 4 (typically -15)
23:16	X	B_H_FILTER_P0BC3: Phase 0B coefficient 3 (typically 102)
15:8	X	B_H_FILTER_P0BC2: Phase 0B coefficient 2 (typically 47)
7:3	X	B_H_FILTER_P0BC1: Phase 0B coefficient 1 (typically -11)
2:0	X	B_H_FILTER_P0BC0: Phase 0B coefficient 0 (typically 2)

25.11.15 DC_WINC_B_H_FILTER_P0C_0

Window B Horizontal Filter phase 0C

Offset: 0x60d │ Byte Offset: 0x1834 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P0CC5: Phase 0C coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P0CC4: Phase 0C coefficient 4 (typically -13)
23:16	X	B_H_FILTER_P0CC3: Phase 0C coefficient 3 (typically 109)
15:8	X	B_H_FILTER_P0CC2: Phase 0C coefficient 2 (typically 37)
7:3	X	B_H_FILTER_P0CC1: Phase 0C coefficient 1 (typically -9)
2:0	X	B_H_FILTER_P0CC0: Phase 0C coefficient 0 (typically 2)

25.11.16 DC_WINC_B_H_FILTER_P0D_0

Window B Horizontal Filter phase 0D

Offset: 0x60e │ Byte Offset: 0x1838 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P0DC5: Phase 0D coefficient 5 (typically 2)
28:24	X	B_H_FILTER_P0DC4: Phase 0D coefficient 4 (typically -11)
23:16	X	B_H_FILTER_P0DC3: Phase 0D coefficient 3 (typically 115)
15:8	X	B_H_FILTER_P0DC2: Phase 0D coefficient 2 (typically 27)
7:3	X	B_H_FILTER_P0DC1: Phase 0D coefficient 1 (typically -7)
2:0	X	B_H_FILTER_P0DC0: Phase 0D coefficient 0 (typically 2)

25.11.17 DC_WINC_B_H_FILTER_P0E_0

Window B Horizontal Filter phase 0E

Offset: 0x60f │ Byte Offset: 0x183c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P0EC5: Phase 0E coefficient 5 (typically 1)
28:24	X	B_H_FILTER_P0EC4: Phase 0E coefficient 4 (typically -8)
23:16	X	B_H_FILTER_P0EC3: Phase 0E coefficient 3 (typically 122)
15:8	X	B_H_FILTER_P0EC2: Phase 0E coefficient 2 (typically 17)
7:3	X	B_H_FILTER_P0EC1: Phase 0E coefficient 1 (typically -5)
2:0	X	B_H_FILTER_P0EC0: Phase 0E coefficient 0 (typically 1)

25.11.18 DC_WINC_B_H_FILTER_P0F_0

Window B Horizontal Filter phase 0F

Offset: 0x610 │ Byte Offset: 0x1840 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	B_H_FILTER_P0FC5: Phase 0F coefficient 5 (typically 1)
28:24	X	B_H_FILTER_P0FC4: Phase 0F coefficient 4 (typically -4)
23:16	X	B_H_FILTER_P0FC3: Phase 0F coefficient 3 (typically 124)
15:8	X	B_H_FILTER_P0FC2: Phase 0F coefficient 2 (typically 8)
7:3	X	B_H_FILTER_P0FC1: Phase 0F coefficient 1 (typically -2)
2:0	X	B_H_FILTER_P0FC0: Phase 0F coefficient 0 (typically 1)

25.11.19 DC_WINC_B_CSC_YOF_0

Window B CSC Y Offset

Color Space Conversion coefficients.

The CSC can be used for YUV to RGB conversion with brightness and hue/saturation control.

The CSC can only be enabled for window B controlled by CSC_ENABLE register bits.

For Y color, the Y offset is applied first and saturation (clipping) is performed immediately after the Y offset is applied.

R = sat(KYRGB * sat(Y + YOF) + KUR * U + KVR * V)

G = sat(KYRGB * sat(Y + YOF) + KUG * U + KVG * V)

B = sat(KYRGB * sat(Y + YOF) + KUB * U + KVB * V)

Saturation and rounding is performed in the range of 0 to 255 for the above equations.

Typical values are:

YOF = -16.000, KYRGB = 1.1644

KUR = 0.0000, KVR = 1.5960

KUG = -0.3918, KVG = -0.8130

KUB = 2.0172, KVB = 0.0000

KUR and KVB are typically 0.0000 but they may be programmed non-zero for hue rotation.

The CSC can also take RGB input, in which case YOF, KVB, KUG, KUR should be programmed to 0, and KYRGB will be forced to 0 by the hardware for generating R and B. KYRGB will not be forced to 0 for generating G. KVR, KYRGB, and KUB can be programmed to 1.0 or used as gain control for R, G, B, correspondingly.

Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Offset: 0x611 | Byte Offset: 0x1844 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_CSC_YOF: Y Offset in s.7.0 format

25.11.20 DC_WINC_B_CSC_KYRGB_0

Window B CSC Y Coefficient (gain) for RGB

Offset: 0x612 | Byte Offset: 0x1848 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	B_CSC_KYRGB: Y Gain for R, G, B colors in 2.8 format

25.11.21 DC_WINC_B_CSC_KUR_0

Window B CSC U coefficient for R

Offset: 0x613 | Byte Offset: 0x184c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	B_CSC_KUR: U coefficients for R in s.2.8 format

25.11.22 DC_WINC_B_CSC_KVR_0

Window B CSC V coefficient for R

Offset: 0x614 | Byte Offset: 0x1850 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	B_CSC_KVR: V coefficients for R in s.2.8 format

25.11.23 DC_WINC_B_CSC_KUG_0

Window B CSC U coefficient for G

Offset: 0x615 | Byte Offset: 0x1854 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	B_CSC_KUG: U coefficients for G in s.1.8 format

25.11.24 DC_WINC_B_CSC_KVG_0

Window B CSC V coefficient for G

Offset: 0x616 | Byte Offset: 0x1858 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	B_CSC_KVG: V coefficients for G in s.1.8 format

25.11.25 DC_WINC_B_CSC_KUB_0

Window B CSC U coefficient for B

Offset: 0x617 | Byte Offset: 0x185c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	B_CSC_KUB: U coefficients for B in s.2.8 format

25.11.26 DC_WINC_B_CSC_KVB_0

Window B CSC V coefficient for B

Offset: 0x618 | Byte Offset: 0x1860 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	B_CSC_KVB: V coefficients for B in s.2.8 format

25.11.27 DC_WINC_B_V_FILTER_P00_0

Window B Vertical Filter phase 00

Vertical scaling filter coefficients

Vertical scaling filter is a 2-tap filter with 4-bit positional phase.

Coefficients 0 and 1 are 8-bit unsigned values ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2 pixels and coefficient 1 is the multiplier for the later pixel (P1) in the group. The output pixel positional phase is defined as centered in P0 if the positional phase is 0 or proportionally in between P0 and P1 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically therefore coefficient 1 can be calculated from (1 - coefficient 0) and therefore only coefficient 0 is programmed.

For each vertical positional phase, the filter coefficient requires 8 register bits. Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Offset: 0x619 | Byte Offset: 0x1864 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P00C0: Phase 00 coefficient 0 (typically 128)

25.11.28 DC_WINC_B_V_FILTER_P01_0

Window B Vertical Filter phase 01

Offset: 0x61a | Byte Offset: 0x1868 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P01C0: Phase 01 coefficient 0 (typically 120)

25.11.29 DC_WINC_B_V_FILTER_P02_0

Window B Vertical Filter phase 02

Offset: 0x61b | Byte Offset: 0x186c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P02C0: Phase 02 coefficient 0 (typically 112)

25.11.30 DC_WINC_B_V_FILTER_P03_0

Window B Vertical Filter phase 03

Offset: 0x61c | Byte Offset: 0x1870 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P03C0: Phase 03 coefficient 0 (typically 104)

25.11.31 DC_WINC_B_V_FILTER_P04_0

Window B Vertical Filter phase 04

Offset: 0x61d | Byte Offset: 0x1874 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P04C0: Phase 04 coefficient 0 (typically 96)

25.11.32 DC_WINC_B_V_FILTER_P05_0

Window B Vertical Filter phase 05

Offset: 0x61e | Byte Offset: 0x1878 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P05C0: Phase 05 coefficient 0 (typically 88)

25.11.33 DC_WINC_B_V_FILTER_P06_0

Window B Vertical Filter phase 06

Offset: 0x61f | Byte Offset: 0x187c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P06C0: Phase 06 coefficient 0 (typically 80)

25.11.34 DC_WINC_B_V_FILTER_P07_0

Window B Vertical Filter phase 07

Offset: 0x620 | Byte Offset: 0x1880 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P07C0: Phase 07 coefficient 0 (typically 72)

25.11.35 DC_WINC_B_V_FILTER_P08_0

Window B Vertical Filter phase 08

Offset: 0x621 | Byte Offset: 0x1884 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P08C0: Phase 08 coefficient 0 (typically 64)

25.11.36 DC_WINC_B_V_FILTER_P09_0

Window B Vertical Filter phase 09

Offset: 0x622 | Byte Offset: 0x1888 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P09C0: Phase 09 coefficient 0 (typically 56)

25.11.37 DC_WINC_B_V_FILTER_P0A_0

Window B Vertical Filter phase 0A

Offset: 0x623 | Byte Offset: 0x188c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P0AC0: Phase 0A coefficient 0 (typically 48)

25.11.38 DC_WINC_B_V_FILTER_P0B_0

Window B Vertical Filter phase 0B

Offset: 0x624 | Byte Offset: 0x1890 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P0BC0: Phase 0B coefficient 0 (typically 40)

25.11.39 DC_WINC_B_V_FILTER_P0C_0

Window B Vertical Filter phase 0C

Offset: 0x625 | Byte Offset: 0x1894 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P0CC0: Phase 0C coefficient 0 (typically 32)

25.11.40 DC_WINC_B_V_FILTER_P0D_0

Window B Vertical Filter phase 0D

Offset: 0x626 | Byte Offset: 0x1898 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P0DC0: Phase 0D coefficient 0 (typically 24)

25.11.41 DC_WINC_B_V_FILTER_P0E_0

Window B Vertical Filter phase 0E

Offset: 0x627 | Byte Offset: 0x189c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P0EC0: Phase 0E coefficient 0 (typically 16)

25.11.42 DC_WINC_B_V_FILTER_P0F_0

Window B Vertical Filter phase 0F

Offset: 0x628 | Byte Offset: 0x18a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	B_V_FILTER_P0FC0: Phase 0F coefficient 0 (typically 8)

The registers under DC_WIN are double buffered

25.11.43 DC_WIN_B_WIN_OPTIONS_0

Window B Options

Class: Display Window Settings

Display Window B parameters

Offset: 0x700 | Byte Offset: 0x1c00 | Read/Write: R/W | Reset: 0x00000000 (0bx0xxxxxxxxxxxx0x0xxxxxxxx0xxxx)

Bit	Reset	Description
30	0x0	B_WIN_ENABLE: Window B Window enable 0 = DISABLE 1 = ENABLE
22	X	B_YUV_RANGE_EXPAND: Window B Enable range expansion in the cases where RANGEREDFRM is 1 from mpd. Formula: Y = clip((Y-128)*2 + 128); Cb = clip((Cb-128)*2 + 128); Cr = clip((Cr-128)*2 + 128); where clip() function clips between 0 and 255. 0 = DISABLE 1 = ENABLE
20	X	B_DV_ENABLE: Window B Digital Vibrance Enable 0 = DISABLE 1 = ENABLE
18	X	B_CSC_ENABLE: Window B Color Space Conversion Enable. This controls the color space conversion and should be enabled for YCbCr/YUV color modes for conversion to B8G8R8 and for hue and saturation control. This can also be used for gain control for RGB color modes 0 = DISABLE 1 = ENABLE
16	0x0	B_CP_ENABLE: Window B Color Palette Enable. This controls the color palette and should be enabled for palletized color modes. For non-palletized color modes, the color palette can be enabled for gamma correction. 0 = DISABLE 1 = ENABLE
14	0x0	B_V_FILTER_UV_ALIGN: Window B V Filter UV Alignment. This is effective only when vertical scaling filter is enabled and only on these formats YCbCr420P, YUV420P, YCbCr422R, YUV422R, YCbCr422RA YUV422RA. When UV alignment is enabled, the chroma components are aligned to the even number of luma component lines. When disabled the chroma components are aligned to half a pixel below the corresponding even number of luma component lines. It is usually disabled unless the incoming video stream specifically indicates otherwise. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	X	B_V_FILTER_OPTIMIZE: Window B V Filter Optimization. This is effective only when vertical scaling filter is enabled. This can be used (enabled) to temporarily disable the vertical scaling filter when the vertical scaling DDA fraction is zero. In this case the next line is not fetched from memory to save bandwidth and power. This feature cannot be used in 420P/422R/422RA formats. 0 = DISABLE 1 = ENABLE
10	X	B_V_FILTER_ENABLE: Window B V Filter Enable. This controls V scaling filter and is effective only for non-palletized color modes. If V filter is disabled, only one line is read from memory for each output line. NOTE: This feature is not supported. This bit is ignored and is a don't care. 0 = DISABLE 1 = ENABLE
8	X	B_H_FILTER_ENABLE: Window B H Filter Enable. This controls H scaling filter and is effective only for non-palletized color modes. 0 = DISABLE 1 = ENABLE
6	X	B_COLOR_EXPAND: Window B 12/15/16/18-to-24 bpp color expansion. This bit should be enabled only for 12-bpp B4G4R4A4, 15-bpp B5G5R5A, 16-bpp B5G6R5, 18-bpp B6G6R6 color modes. If enabled the color conversion is performed prior to horizontal scaling. 0 = DISABLE 1 = ENABLE
4	0x0	B_SCAN_COLUMN: Window B Scanning direction. 0= Scan in horizontal direction (for 0 or 180 degrees) 1= Scan in vertical direction (for 90 or 270 degrees) 0 = DISABLE 1 = ENABLE
2	X	B_V_DIRECTION: Window B Vertical (Y) drawing Direction 0 = INCREMENT 1 = DECREMENT
0	X	B_H_DIRECTION: Window B Horizontal (X) drawing Direction 0 = INCREMENT 1 = DECREMENT

25.11.44 DC_WIN_B_BYTE_SWAP_0

Window B Byte Swap

Offset: 0x701 │ Byte Offset: 0x1c04 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2:0	X	B_BYTE_SWAP: Window B Byte Swap. This controls byte swap of frame data read from memory prior to any data processing in the display module. 000= no byte swap (3 2 1 0) 001= byte swap for each 2-byte word (2 3 0 1) 010= byte swap for each 4-byte word (0 1 2 3) 011= word swap for each 4-byte word (1 0 3 2) 100= byte0 swapped with byte2 (3 0 1 2) 101= left shift every byte (2 1 0 3) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW 4 = SWAP02 5 = SWAPLEFT

25.11.45 DC_WIN_B_COLOR_DEPTH_0

Window B Color Depth for YCbCr data format. Cb and Cr are 8-bit unsigned values. For YUV data format, U and V are 8-bit signed values. YCbCr422R is similar to YCbCr422P, but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P, but the U and V are shared vertically. YCbCr422RA is the same as YCbCr422R in memory, and YUV422RA is the same as YUV422R in memory but while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixel is not used (discarded) by the display pipeline. B6x2G6x2R6x2A8 is similar to B8G8R6A8 but with the 2 LSBs zeroed out. R6x2G6x2B6x2A8 is similar to R8G8B6A8 but with the 2 LSBs zeroed out.

Some formats have NVCF_ or T_ prefix aliases for NVIDIA surface format naming convention, wherein left-to-right component names correspond to MSB to LSB bits within a word. In old names without NVCF_ or T_, left-to-right component names generally map LSB to MSB within a word.

Offset: 0x703 │ Byte Offset: 0x1c0c │ Read/Write: R/W │ Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx001100)

Bit	Reset	Description
5:0	B8G8R8A8	B_COLOR_DEPTH: Window B Color Depth. Supported color depths are: P8 = 8-bpp (palletized) B4G4R4A4 = 12-bpp B4G4R4 B5G5R5A = 15-bpp B5G5R5 AB5G5R5 = 15-bpp B5G5R5 B5G6R5 = 16-bpp B5G6R5 B8G8R8A8 = 32-bpp B8G8R8A8 R8G8B8A8 = 32-bpp R8G8B8A8 YCbCr422 = 16-bpp YCbCr422 packed YUV422 = 16-bpp YUV422 YCbCr420P = 16-bpp YCbCr420 planar YUV420P = 16-bpp YUV420 planar YCbCr422P = 16-bpp YCbCr422 planar YUV422P = 16-bpp YUV422 planar YCbCr422R = 16-bpp YCbCr422 rotated planar YUV422R = 16-bpp YUV422 rotated planar YCbCr422RA= 16-bpp YCbCr422 rotated planar with chroma averaging YUV422RA = 16-bpp YUV422 rotated planar with chroma averaging Reserved values: 0,1,2,14,15,24,25. 3 = T_P8, P8 4 = T_A4R4G4B4, B4G4R4A4 5 = T_A1R5G5B5, B5G5R5A 6 = T_R5G6B5, B5G6R5 7 = T_R5G5B5A1, AB5G5R5 12 = T_A8R8G8B8, B8G8R8A8 13 = T_A8B8G8R8, R8G8B8A8 16 = T_U8_Y8__V8_Y8, CbYCrY422, YCbCr422 17 = T_U8_Y8__V8_Y8_TRUE, UYVY422, YUV422, TRUE_U8_Y8__V8_Y8 18 = T_Y8___U8___V8_N420, YCbCr420P 19 = T_Y8___U8___V8_N420_TRUE, YUV420P, TRUE_Y8___U8___V8_N420 20 = T_Y8___U8___V8_N422, YCbCr422P 21 = T_Y8___U8___V8_N422_TRUE, YUV422P, TRUE_Y8___U8___V8_N422 22 = T_Y8___U8___V8_N422R, YCbCr422RP, YCbCr422R 23 = T_Y8___U8___V8_N422R_TRUE, YUV422RP, YUV422RTRUE_Y8___U8___V8_N422R (new format, reserved for future use) 26 = T_Y8, Y8 27 = T_A4B4G4R4, R4G4B4A4 28 = T_A1B5G5R5, R5G5B5A 29 = T_B5G5R5A1, AR5G5B5 30 = T_X1R5G5B5, B5G5R5X1 31 = T_R5G5B5X1, X1B5G5R5 32 = T_X1B5G5R5, R5G5B5X1

Bit	Reset	Description
		33 = T_B5G5R5X1, X1R5G5B5 34 = T_B5G6R5, R5G6B5 35 = T_B8G8R8A8 36 = T_R8G8B8A8 37 = T_X8R8G8B8, B8G8R8X8 38 = T_X8B8G8R8, R8G8B8X8 39 = T_A8Y8U8V8 40 = T_V8U8Y8A8 41 = T_Y8___U8___V8_N444, YCbCr444P 42 = T_Y8___U8V8_N420, YCrCb420SP 43 = T_Y8___V8U8_N420, YCbCr420SP 44 = T_Y8___U8V8_N422, YCrCb422SP 45 = T_Y8___V8U8_N422, YCbCr422SP 46 = T_Y8___U8V8_N422R, YCrCb422RSP 47 = T_Y8___V8U8_N422R, YCbCr422RSP 48 = T_Y8___U8V8_N444, YCrCb444SP 49 = T_Y8___V8U8_N444, YCbCr444SP 50 = T_A8Y8U8V8_TRUE, TRUE_A8Y8U8V8 51 = T_V8U8Y8A8_TRUE, TRUE_V8U8Y8A8 52 = T_Y8___U8___V8_N444_TRUE, YUV444P, TRUE_Y8___U8___V8_N444 53 = T_Y8___U8V8_N420_TRUE, YVU420SP, TRUE_Y8___U8V8_N420 54 = T_Y8___V8U8_N420_TRUE, YUV420SP, TRUE_Y8___V8U8_N420 55 = T_Y8___U8V8_N422_TRUE, YVU422SP, TRUE_Y8___U8V8_N422 56 = T_Y8___V8U8_N422_TRUE, YUV422SP, TRUE_Y8___V8U8_N422 57 = T_Y8___U8V8_N422R_TRUE, YVU422RSP, TRUE_Y8___U8V8_N422R 58 = T_Y8___V8U8_N422R_TRUE, YUV422RSP, TRUE_Y8___V8U8_N422R 59 = T_Y8___U8V8_N444_TRUE, YUV444SP, TRUE_Y8___U8V8_N444 60 = T_Y8___V8U8_N444_TRUE, YVU444SP, TRUE_Y8___V8U8_N444

25.11.46 DC_WIN_B_POSITION_0

Window B Position

This register defines H position and size of Window B after scaling (if there is any)

Offset: 0x704 │ Byte Offset: 0x1c10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	B_V_POSITION: Window B V Position. This is specified with respect to the top edge of active display area.
12:0	X	B_H_POSITION: Window B H Position. This is specified with respect to the left edge of active display area.

25.11.47 DC_WIN_B_SIZE_0

Window B Size

This register defines V position and size of Window B after scaling (if there is any)

Note: programming on window size should guarantee the whole window is inside the active area; otherwise extra rows/columns will be fetched and discarded, affecting performance.

Offset: 0x705 │ Byte Offset: 0x1c14 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	B_V_SIZE: Window B V Size (lines). This is the vertical size after scaling.
12:0	X	B_H_SIZE: Window B H Size (pixels). This is the horizontal size after scaling.

25.11.48 DC_WIN_B_PRESCALED_SIZE_0

Window B Pre-scaled Size

This register defines Window B pre-scaled size.

The H pre-scaled size is needed to determine how many bytes to fetch from memory per line and this parameter must be programmed exactly as needed taking into account the scaling factor. For planar YUV or YCbCr data formats, this parameter refer to the H pre-scaled of the Y plane.

The total number of lines to be fetched from memory is determined by post-scale V size but V pre-scaled size is needed to 'clamp' the last valid line if the vertical DDA is exactly or slightly beyond the specified V pre-scaled size.

H pre-scaled size ideally should be in terms of pixels but then hardware needs to convert this precisely to bytes to determine the amount of data to request from memory.

Offset: 0x706 │ Byte Offset: 0x1c18 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	B_V_PRESCALED_SIZE: Window B V Pre-scaled Size (lines). In 420P/422R/422RA formats, it must be even.
14:0	X	B_H_PRESCALED_SIZE: Window B H Pre-scaled Size (bytes). In 420P and 422P formats, it must be even.

25.11.49 DC_WIN_B_H_INITIAL_DDA_0

Window B H Initial DDA

Design Note: the first pixel of pre-scaled image is always used to output the first pixel so essentially this is the same as forcing the H Initial DDA integer portion to 1 initially even though user typically programs this to 0. If it makes the implementation easier, it is possible to force software to program the Initial DDA integer portion to 1. Similarly with the V Initial DDA.

Offset: 0x707 │ Byte Offset: 0x1c1c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	B_H_INITIAL_DDA: Window B H Initial DDA (4.12). This is typically programmed to 0.0

25.11.50 DC_WIN_B_V_INITIAL_DDA_0

Window B V Initial DDA

Offset: 0x708 │ Byte Offset: 0x1c20 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	B_V_INITIAL_DDA: Window B V Initial DDA (4.12). This is typically programmed to 0.0 for both non-interlaced and interlaced sources.

25.11.51 DC_WIN_B_DDA_INCREMENT_0

Window B DDA Increment

The DDA increment is typically calculated by dividing (Pre-scaled size in pixels - 1) by (Post-scaled size in pixels - 1). The result should be rounded up and expressed as 4.12 format (4-bit integer and 12-bit fraction). For non-filtered images, this

value can be slightly larger so that it is not missing the last row/column. Reference programming values (H and V should be calculated separately depending on filter on/off and sizes):

- Filter on: min(round((prescaled_size_in_pixels - 1) * 0x1000 / (post_scaled_size_in_pixels - 1)), MAX)
- Filter off: min(round(prescaled_size_in_pixels * 0x1000 / (post_scaled_size_in_pixels - 1) - 0.5), MAX)

Where the value of MAX is as follows:

For V_DDA_INCREMENT: 15.0 (0xF000)

For H_DDA_INCREMETN: 4.0 (0x4000) for 4 Bytes/pix formats.

8.0 (0x8000) for 2 Bytes/pix formats.

They are theoretically the biggest values that guarantee not displaying beyond an image boundary.

If the DDA increment is less than 1.0 then the image will be upscaled.If the DDA increment is more than 1.0 then the image will be downscaled.

Offset: 0x709 │ Byte Offset: 0x1c24 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	B_V_DDA_INCREMENT: Window B Vertical DDA Increment (4.12) This should be set to 1.0 if there is no scaling. Maximum value is 15.0 regardless of the number of bytes per pixel.
15:0	X	B_H_DDA_INCREMENT: Window B Horizontal DDA Increment (4.12) This should be set to 1.0 if there is no scaling. The maximum value for downscaling depends on the number of bytes per pixel. For 4-byte/pixel modes (32-bpp) the maximum value is 4.0 and for all other modes the maximum value is 8.0.

25.11.52 DC_WIN_B_LINE_STRIDE_0

Window B Line Stride

Offset: 0x70a │ Byte Offset: 0x1c28 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	B_UV_LINE_STRIDE: Window B Line Stride for Chroma This is stride (in bytes) for planar YUV or YCbCr data formats for the chroma plane, with the restriction that it must be programmed to be multiples of 4 (16 if tiled or in horizontal flipping) This is not used (ignored) for other non-planar data formats.
15:0	X	B_LINE_STRIDE: Window B Line Stride This is stride (in bytes) for all non-planar data formats. If the memory surface is tiled, the stride needs to be a multiple of 16. If H_DIRECTION of window B is set to DECREMENT, the stride also needs to be a multiple of 16. For planar YUV or YCbCr data formats, this is stride (in bytes) for the luma plane with the restriction that it must be multiples of 8 (16 if tiled or in horizontal flipping). For tiled surface, this value may affect starting address of a window. Refer to the comment of START_ADDR for detail.

25.11.53 DC_WIN_B_BUFFER_ADDR_MODE_0

Memory Controller Tiling definitions

Offset: 0x70d │ Byte Offset: 0x1c34 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxxxxxxxxxx0)

Bit	Reset	Description
16	0x0	B_UV_TILE_MODE: Window B Memory surface tiling mode. For YUV planar pixel format, this specifies tiling mode for the U/V plane.

Bit	Reset	Description
		0 = LINEAR 1 = TILED
0	0x0	B_TILE_MODE: Window B Memory surface tiling mode. For YUV planar pixel format, this specifies tiling mode for the Y plane. 0 = LINEAR 1 = TILED

25.11.54 DC_WIN_B_DV_CONTROL_0

Window B Digital Vibrance Control

If enabled, Digital Vibrance is applied after H and V scaling and after color palette or color space conversion logic but before color keying multiplexer and before cursor multiplexer.

- After DV, new R = R + (2R - G - B) * FR, where FR is fraction from 0 to 7/8
- After DV, new G = G + (2G - R - B) * FG, where FG is fraction from 0 to 7/8
- After DV, new B = B + (2B - R - G) * FB, where FB is fraction from 0 to 7/8

Offset: 0x70e │ Byte Offset: 0x1c38 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18:16	X	B_DV_CONTROL_B: Digital Vibrance control for B
10:8	X	B_DV_CONTROL_G: Digital Vibrance control for G
2:0	X	B_DV_CONTROL_R: Digital Vibrance control for R

25.11.55 DC_WIN_B_BLEND_NOKEY_0

Blend Control for this window area where color key is enabled but the pixel color is not within the color key range (color key not match). This is valid for all overlapping condition but only if there is no overlap with other window with higher priority color key enabled and color key not match.

Class: Display Color Keying and Blending

Color keying and blending of the display windows are done prior to cursor blending.

Cursor always goes on top of the blended windows. Blending is controlled independently on each possible overlap area of the display windows. If 3 windows are enabled there are 7 possible overlap area combinations. For every window in each overlap area combination, color key can be disabled or enabled. Also, for every window in each overlap area combination, there is a corresponding window blend control parameter and a window blend weight parameter. The window blend control parameter is always effective but the window blend weight is not always used. The window blend weight can also be derived from pixel alpha value or from the reverse of other overlapping windows weight.

Color keying has the highest priority for display window blending. Color key consists of a range of color which is searched independently for each windows. If more than 1 windows color key is enabled, then Window A color key has the highest priority, followed by Window B color key, and then followed by Window C color key. Two sets of color key ranges (Color Key 0 and Color Key 1) can be defined; they are shared for all windows. It is possible to use both color key sets for the same window or for two separate windows.

The two sets of color key ranges should not overlap. If they do, the overlapped colors are treated as if they are part of Color Key 0 and not part of Color Key 1.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for any overlap condition and the window pixel is not within the color key range (key not match), then the window pixel will not be blended with other overlapping window pixels but it will be weighted. The weight is not dependent on the overlap condition it is controlled by the same set of parameters for all overlapping condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for a particular overlap condition and the window pixel is within the color key range (key match), then the window pixel will be weighted and blended with other overlapping pixels and this is controlled separately for each overlap condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is disabled then the window pixel will be blended with other overlapping pixels and this is controlled separately for each overlap condition.

Display Color Key Parameters

For B4G4R4A4, B5G6R5A, B5G6R5 modes, the color key should be compared prior to color conversion to 24-bpp and unused least significant bits of the pixel are filled with zeros.

For palletized mode, the color key is compared prior to the color palette, and the palletized color is compared against the green color key values/mask.

For YUV mode, U and V are offset by +128 before performing the color key comparison.

In all cases, the color key is compared prior to the horizontal/vertical scaling filter and prior to digital vibrance control. Both upper and lower values are inclusive.

Offset: 0x70f │ Byte Offset: 0x1c3c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	B_BLEND_WEIGHT1_NOKEY: Window blend weight 1 for color key not match areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	B_BLEND_WEIGHT0_NOKEY: Window blend weight 0 for color key not match areas. For alpha weight, this is used for 1-bit alpha with value of 0.
0	X	B_BLEND_CONTROL_NOKEY: Window blend control for color key not match areas. 0 = Fix weight using window blend weight 0 for color key not matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT

25.11.56 DC_WIN_B_BLEND_1WIN_0

Blend Control for this window area where it does not overlap with other windows.

Offset: 0x710 │ Byte Offset: 0x1c40 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	B_BLEND_WEIGHT1_1WIN: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	B_BLEND_WEIGHT0_1WIN: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
2	X	B_BLEND_CONTROL_1WIN: Window blend control in area where it does not overlap with other windows and either color key disabled or color key enabled with key matched.

Bit	Reset	Description
		0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0, window blend weight 0 is used; if alpha is 1, window blend weight 1 is used. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT
1:0	X	B_CKEY_ENABLE_1WIN: Window color key enable. 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

25.11.57 DC_WIN_B_BLEND_2WIN_A_0

Blend Control for this window area that overlaps with window A only.

Offset: 0x711 │ Byte Offset: 0x1c44 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	B_BLEND_WEIGHT1_2WIN_A: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	B_BLEND_WEIGHT0_2WIN_A: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	B_BLEND_CONTROL_2WIN_A: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 3 = Premultipled weight; in this case, the window blend weight is 1. But other windows that use DEPENDENT_WEIGHT get (1 - alpha value),like ALPHA_WEIGHT. This is only valid if the window color format includes an alpha value. This allows a blending equation like: result = win_b * 1 + (1 - win_b_alpha) * win_c 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT 3 = PREMULT_WEIGHT
1:0	X	B_CKEY_ENABLE_2WIN_A: Window color key enable. 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY

Bit	Reset	Description
		1 = CKEY0 2 = CKEY1 3 = CKEY01

25.11.58 DC_WIN_B_BLEND_2WIN_C_0

Blend Control for this window area that overlaps with window C only.

Offset: 0x712 | Byte Offset: 0x1c48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	B_BLEND_WEIGHT1_2WIN_C: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	B_BLEND_WEIGHT0_2WIN_C: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	B_BLEND_CONTROL_2WIN_C: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 3 = Premultipled weight; in this case, the window blend weight is 1. But other windows that use DEPENDENT_WEIGHT get (1 - alpha value), like ALPHA_WEIGHT. This is only valid if the window color format includes an alpha value. This allows a blending equation like: result = win_a * 1 + (1 - win_a_alpha) * win_b 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT 3 = PREMULT_WEIGHT
1:0	X	B_CKEY_ENABLE_2WIN_C: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

25.11.59 DC_WIN_B_BLEND_3WIN_AC_0

Blend Control for this window area that overlaps with windows A & C only.

Offset: 0x713 | Byte Offset: 0x1c4c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	B_BLEND_WEIGHT1_3WIN_AC: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and

Bit	Reset	Description
		alpha value of 1.
15:8	X	B_BLEND_WEIGHT0_3WIN_AC: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	B_BLEND_CONTROL_3WIN_AC: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 3 = Premultipled weight; in this case, the window blend weight is 1. But other windows that use DEPENDENT_WEIGHT get (1 - alpha value), like ALPHA_WEIGHT. This is only valid if the window color format includes an alpha value 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT 3 = PREMULT_WEIGHT
1:0	X	B_CKEY_ENABLE_3WIN_AC: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

25.11.60 DC_WIN_B_GLOBAL_ALPHA_0

Window B Global Alpha

When enabled, per pixel alpha values are scaled by GLOBAL_ALPHA before blending: a' = a * GLOBAL_ALPHA/255. When per pixel alpha is not used in blending, global alpha has no affect.

Note: 1- and 4-bits per pixel alpha are quantized back to 1 and 4 bits, respectively, after scaling. For the 1-bit alpha case, finer scaling can be achieved by changing BLEND_WEIGHT* instead.

Offset: 0x715 │ Byte Offset: 0x1c54 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxxxxxxxxxx)

Bit	Reset	Description
16	DISABLE	B_GLOBAL_ALPHA_ENABLE: When enabled, color palette *must* also be enabled and programmed. 0 = DISABLE 1 = ENABLE
7:0	X	B_GLOBAL_ALPHA: Global alpha: 0..255 map to 0.0 – 1.0. i.e., 255 yields no-operation. For power, better to set GLOBAL_ALPHA_ENABLE=DISABLE when GLOBAL_ALPHA=255.

25.12 WINBUF_B Registers

The registers under DC_WINBUF are triple-buffered.

25.12.1 DC_WINBUF_B_START_ADDR_0

Window B Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specifies the address offsets of the pixel at the beginning of the window, with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see ”For linear address mode” in Programming Restrictions below.)

Note that "beginning of the window" here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting Address Calculation

These formulae are the same for both tiled and linear modes. However, for linear mode, the addresses calculated are real physical addresses, but for tiled mode, these addresses will be translated to the real physical addresses by the memory controller client before used.

- When a window is host triggered, the starting address of a window is calculated as follows by hardware:
  - non-YUV-planar modes:

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar modes:

    Y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.
- When a window is non-host triggered, starting address of a window buffer is calculated as below:
  - non-YUV-planar mode:

    starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar mode:

    Y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

    buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

- For tiled address mode:

  Image surface can only aligned to multiples of 256, thus the following restrictions.

  - START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.
  - BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256
  - LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16
  - ADDR_H_OFFSET needs to be even in YUV planar format, or a multiple of bytes per pixel in other formats. If H_DIRECTION=DECREMENT, however, it should point to last valid byte, which is an odd offset.
  - ADDR_V_OFFSET needs to be multiple of 2 in YUV planar format, unless H_DIRECTION=DECREMENT, but with no restrictions on other color formats.

- For linear address mode:

  Image surface can be aligned 2, 4 or 8 bytes, depending on the color formats.

  As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

  When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formulae in Starting Address Calculation above still hold)

  - For all formats:

    START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.
  - For 16-bpp formats,

    (START_ADDR+H_OFFSET) needs to be a multiple of 2.
  - For 32-bpp formats,

    (START_ADDR+H_OFFSTE) needs to be a multiple of 4.
  - For YUV planar formats:

    BUF_STRIDE, UV_BUF_STRIDE:

    BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

    or as a stricter constraint: BUF_STRIDE be multiple of 8, UV_BUF_STRIDE be a multiple of 4.

    LINE_STRIDE, UV_LINE_STRIDE:

    LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

    LINE_STRIDE needs to be multiple of 8, UV_LINE_STRIDE needs to be a multiple of 4.

    ADDR_H_OFFSET: Needs to be even unless H_DIRECTION=DECREMENT, in which case should point to the last byte pixel, which is at an odd offset.

    ADDR_V_OFFSET: Needs to be even unless V_DIRECTION=DECREMENT, in which case should point to the last valid line, which is at an odd offset.

- Memory allocation for non-host triggered case

  When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because the display will use the same luma or chroma line strides for all lines in the frame.

  Also buffer wraparound must not occur in the middle of the displayed part of the frame. The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not switched until frame end flag is active. In the case where one buffer corresponds to one frame then frame start and frame end flag are active every time a buffer index is sent.

Offset: 0x800 | Byte Offset: 0x2000 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR: Window B Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

25.12.2 DC_WINBUF_B_START_ADDR_NS_0

Window B Shadowed Start Address

Offset: 0x801 | Byte Offset: 0x2004 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_NS: Window B Shadowed Start Address. This is ARM set shadow of Start Address.

25.12.3 DC_WINBUF_B_START_ADDR_U_0

Window B Start Address for U plane

Offset: 0x802 | Byte Offset: 0x2008 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_U: Window B Start Address for U plane. This is a byte address.

25.12.4 DC_WINBUF_B_START_ADDR_U_NS_0

Window B Shadowed Start Address for U plane

Offset: 0x803 | Byte Offset: 0x200c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_U_NS: Window B Shadowed Start Address for U plane. This is ARM set shadow register of U start address

25.12.5 DC_WINBUF_B_START_ADDR_V_0

Window B Start Address for V plane

Offset: 0x804 | Byte Offset: 0x2010 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_START_ADDR_V: Window B Start Address for V plane. This is a byte address.

25.12.6 DC_WINBUF_B_START_ADDR_V_NS_0

Window B Shadowed Start Address for V plane

Offset: 0x805 | Byte Offset: 0x2014 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description

31:0	X	B_START_ADDR_V_NS: Window B Shadowed Start Address for V plane. This is ARM set shadow register of U start address

25.12.7 DC_WINBUF_B_ADDR_H_OFFSET_0

Window B Horizontal address offset

Offset: 0x806 | Byte Offset: 0x2018 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_ADDR_H_OFFSET: Window B Horizontal address offset. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by hardware.

25.12.8 DC_WINBUF_B_ADDR_H_OFFSET_NS_0

Window B Shadowed Horizontal address offset

Offset: 0x807 | Byte Offset: 0x201c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_ADDR_H_OFFSET_NS: Window B Shadowed Horizontal address offset. This is ARM set shadow of ADDR_H_OFFSET

25.12.9 DC_WINBUF_B_ADDR_V_OFFSET_0

Window B Vertical address offset

Offset: 0x808 | Byte Offset: 0x2020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_ADDR_V_OFFSET: Window B Vertical address offset. This is the vertical coordinate. For YUV planar pixel format, this specifies the vertical coordinate of Y plane. The vertical coordinate of U/V plane is derived by hardware.

25.12.10 DC_WINBUF_B_ADDR_V_OFFSET_NS_0

Window B Shadowed Vertical address offset

Offset: 0x809 | Byte Offset: 0x2024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_ADDR_V_OFFSET_NS: Window B Shadowed Vertical address offset. This is ARM set shadow of ADDR_V_OFFSET.

25.12.11 DC_WINBUF_B_UFLOW_STATUS_0

Window B FIFO Underflow Status Register

Offset: 0x80a | Byte Offset: 0x2028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.
23:0	X	UFLOW_COUNT: Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.

25.12.12 DC_WINBUF_B_SURFACE_WEIGHT_0

Window B Surface Weights

Offset: 0x80c │ Byte Offset: 0x2030 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
6:5	X	B_SURFACE_WEIGHT_V: Window B V Surface Weights 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
4:3	X	B_SURFACE_WEIGHT_U: Window B U or UV Surface Weights 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
2:1	X	B_SURFACE_WEIGHT_Y: Window B Y or packed Surface Weights 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
0	0x0	B_SURFACE_WEIGHT_OVERRIDE: weight override 0 = DISABLE 1 = ENABLE

25.12.13 DC_WINBUF_B_START_ADDR_HI_0

Window B Higher 32 bits of Start Address

Make the start address 64 bits by adding the higher 32 bits below.

Offset: 0x80d │ Byte Offset: 0x2034 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	B_START_ADDR_HI: Window B Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

25.12.14 DC_WINBUF_B_START_ADDR_HI_NS_0

Window B Shadowed Higher 32 bits of Start Address

Offset: 0x80e │ Byte Offset: 0x2038 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	B_START_ADDR_HI_NS: Window B Shadowed Start Address. This is ARM set shadow of Start Address.

25.12.15 DC_WINBUF_B_START_ADDR_HI_U_0

Window B Higher 32 bits of Start Address for U plane

Offset: 0x80f │ Byte Offset: 0x203c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	B_START_ADDR_HI_U: Window B Start Address for U plane. This is a byte address.

25.12.16 DC_WINBUF_B_START_ADDR_HI_U_NS_0

Window B Shadowed Higher 32 bits of Start Address for U plane

Offset: 0x810 │ Byte Offset: 0x2040 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	B_START_ADDR_HI_U_NS: Window B Shadowed Higher 32 bits of Start Address for U plane. This is ARM set shadow register of U start address

25.12.17 DC_WINBUF_B_START_ADDR_HI_V_0

Window B Higher 32 bits of Start Address for V plane

Offset: 0x811 │ Byte Offset: 0x2044 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	B_START_ADDR_HI_V: Window B Higher 32 bits of Start Address for V plane. This is a byte address.

25.12.18 DC_WINBUF_B_START_ADDR_HI_V_NS_0

Window B Shadowed Higher 32 bits of Start Address for V plane

Offset: 0x812 │ Byte Offset: 0x2048 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	B_START_ADDR_HI_V_NS: Window B Shadowed Higher 32 bits of Start Address for V plane. This is ARM set shadow register of U start address

25.12.19 DC_WINBUF_B_UFLOW_CTRL_0

When the underflow debug mode is enabled. Instead of sending out the last valid pixel, DBG_PIXEL is sent out.

Offset: 0x824 | Byte Offset: 0x2090 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	B_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

25.12.20 DC_WINBUF_B_UFLOW_DBG_PIXEL_0

Offset: 0x825 | Byte Offset: 0x2094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	B_UFLOW_DBG_PIXEL

25.12.21 DC_WINBUF_B_UFLOW_THRESHOLD_0

Offset: 0x826 | Byte Offset: 0x2098 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	B_UFLOW_THRESHOLD

25.12.22 DC_WINBUF_B_SPOOL_UP_0

Spool up time configuration for different windows related logic.

SPOOL_UP_CTRL = MAX - This is the current Tegra 4 behavior. This allows for memfetch to start fetching for a window (irrespective of its relative position on the screen), at the very beginning of the vblank of the next frame. Since the spool up time is maximum, this should have the least probability of underflow of the display pipe. However, on the down side, since all three windows would start fetching at the same time, and since display is a high priority client for MC, it has a high potential to starve out other clients.

SPOOL_UP_CTRL = PROGRAMMABLE, SPOOL_UP_DURATION = 1 - This corresponds to Tegra 3 behavior. Since the fetch starts only 1 line prior to the active scan line of a window, this has high probability of causing underflow, but has a lesser impact on the other MC clients.

So, this register programmability programs us a method to modulate the spool up time between the best and worst, if required, when other MC clients are present. For example when GPU/VIC is active, we may not want a very aggressive spool up and program some definite spool up duration. So, spool up time is a general knob, which would be a function of the resolution, bpp, active clients, memory speed, etc.

SPOOL_UP_EDGE:- By default the memfetch start signal generated by the hvtgen logic is one line clock wide.

For SPOOL_UP_CTRL = MAX:

- In this case if SPOOL_UP_EDGE is programmed to NEGEDGE. This ensures that if any act_req is sent during the time vcounter = 0(at the beginning of the frame). The activate value is loaded into the active state and the memfetch would see the latest updated value. The downside of programming it to NEGDGE is that 1 line clock worth of spool up is lost. This is required based upon Tegra 3 compatibility and the way tests are written.
- In this case if SPOOL_UP_EDGE is programmed to POSEDGE. Then the memfetch would start fetching at the very beginning of the frame start pulse (1 line clock wide) and would give maximum spool up time. However, this may give test failure and potential Tegra 3 compatibility issues as described above

FOR SPOOL_UP_CTRL = PROGRAMMABLE:

- Special care must be taken here. If SPOOL_UP_DURATION is set to 1. SPOOL_UP_EDGE must be set to POSEDGE to allow for at least 1 line worth of spool up.
- If SPOOL_UP_DURATION is > 1. Then SPOOL_UP_EDGE can be either POSEDGE or NEGEDGE. If its NEGEDGE, then the effective spool up duration would be 1 line clock less than the SPOOL_UP_DURATION.

Offset: 0x827 | Byte Offset: 0x209c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
28:16	X	B_SPOOL_UP_DURATION
1	0x0	B_SPOOL_UP_EDGE: 0 = NEGEDGE 1 = POSEDGE
0	0x0	B_SPOOL_UP_CTRL: 0 = MAX 1 = PROGRAMMABLE

25.12.23 DC_WINBUF_B_SCALEFACTOR_THRESHOLD_0

Registers for latency allowance scaling. Threshold is in terms of 64 byte atoms. Display memfetch compares its buffer occupancy with the threshold and generates indication to MC if its buffer occupancy is above HWM or below LWM.

Offset: 0x828 | Byte Offset: 0x20a0 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:16	0xffff	B_SF_LWM_THRESHOLD
15:0	0xffff	B_SF_HWM_THRESHOLD

25.12.24 DC_WINBUF_B_LATENCY_THRESHOLD_0

Register for ready for latency event like DVFS. Threshold is in terms of 64 byte atoms. Display memfetch compares its buffer occupancy with the threshold and generates indication to MC if its buffer occupancy is above the threshold.

Offset: 0x829 | Byte Offset: 0x20a4 | Read/Write: R/W | Reset: 0x0000ffff (0bxxxxxxxxxxxxxxxx1111111111111111)

Bit	Reset	Description
15:0	0xffff	B_RDY4LATENCY_THRESHOLD

25.12.25 DC_WINBUF_B_MEMFETCH_DEBUG_STATUS_0

Memfetch Debug Status

Offset: 0x82a | Byte Offset: 0x20a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15	0x0	B_ALIGN_FIFO_NON_IDLE: align FIFO idle at end of frame. 0= align FIFO is idle at end of frame. 1= align FIFO is not idle at end of frame
14	0x0	B_PIPE1_FIFO_NON_IDLE: pipe1 FIFO idle at end of frame. 0= pipe1 FIFO is idle at end of frame. 1= pipe1 FIFO is not idle at end of frame
13	0x0	B_PIPE0_FIFO_NON_IDLE: pipe0 FIFO idle at end of frame. 0= pipe0 FIFO is idle at end of frame. 1= pipe0 FIFO is not idle at end of frame
12	0x0	B_FRAME_FIFO_NON_IDLE: frame FIFO idle at end of frame. 0= frame FIFO is idle at end of frame.

Bit	Reset	Description
		1= frame FIFO is not idle at end of frame
11	0x0	B_SURFACE_FIFO_NON_IDLE: surface FIFO idle at end of frame. 0= surface FIFO is idle at end of frame. 1= surface FIFO is not idle at end of frame
10	0x0	B_TAG_FIFO_NON_IDLE: tag FIFO idle at end of frame. 0= tag FIFO is idle at end of frame. 1= tag FIFO is not idle at end of frame
9	0x0	B_RSP_FIFO_NON_IDLE: Response FIFO idle at end of frame. 0= response FIFO is idle at end of frame. 1= response FIFO is not idle at end of frame
8	0x0	B_REQ_FIFO_NON_IDLE: Request FIFO idle at end of frame. 0= request FIFO is idle at end of frame. 1= request FIFO is not idle at end of frame
7	0x0	B_DP_VALID_NON_IDLE: DP lines valid at end of frame. 0= DP lines are not valid at end of frame. 1= DP lines are valid at end of frame
6	0x0	B_V_SSM_NON_IDLE: V SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
5	0x0	B_U_SSM_NON_IDLE: U SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
4	0x0	B_Y_SSM_NON_IDLE: Y SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
3	0x0	B_DP_POOL_AVAIL: DP thinks there is data in the pool at end of frame. 0= No pool data available at end of frame. 1= pool data available at end of frame
2	0x0	B_POOL_NOT_EMPTY: Status of pool at end of frame. 0= pool is empty at end of frame. 1= pool is not empty at end of frame
1	0x0	B_UNDERFLOW_LINE1: Underflow of line0. 0= No underflow 1= line0 underflowed
0	0x0	B_UNDERFLOW_LINE0: Underflow of line0. 0= No underflow 1= line0 underflowed

25.12.26 DC_WINBUF_B_MEMFETCH_CONTROL_0

Memfetch Control Register

Offset: 0x82b │ Byte Offset: 0x20ac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	B_MEMFETCH_CLK_GATE_OVERRIDE: Disable clock-gating of memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE
0	0x0	B_MEMFETCH_RESET: Reset memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE

25.12.27 DC_WINBUF_B_OCCUPANCY_THROTTLE_0

Offset: 0x82c | Byte Offset: 0x20b0 | Read/Write: R/W | Reset: 0xffff0000 (0b1111111111111111xxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:16	0xffff	B_OCCUPANCY_MAX_THRESHOLD
0	0x0	B_OCCUPANCY_THROTTLE_MODE: 0 = DISABLE 1 = ENABLE

25.12.28 DC_WINBUF_B_SCRATCH_REGISTER_0_0

Offset: 0x82d | Byte Offset: 0x20b4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	B_SCRATCH_REGISTER_0:Scratch register 0

25.12.29 DC_WINBUF_B_SCRATCH_REGISTER_1_0

Offset: 0x82e | Byte Offset: 0x20b8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	B_SCRATCH_REGISTER_1:Scratch register 1

25.13 Window C (WIN_C) Registers

These registers control window C parameters

25.13.1 DC_WINC_C_COLOR_PALETTE_0

Window C Color Palette

This is used for palletized data format (color depth of 8-bpp or less) or for gamma correction for non-palletized data formats (color depth of more than 8 bpp). Each window has its own color palette which consists of three 256x8 register file which can be written by host and indexed (read) by the window.

For palletized data format less than 8 bpp, the pixel data is aligned to least significant bits of the palette index (address) and the remaining upper bits are filled with the corresponding bits of the Palette Color Extension. For example, for 4-bpp mode, the pixel data occupies bits 3:0 of the palette index and bits 7:4 of the palette index are set to bits 7:4 of the Palette Color Extension. Host read is assumed to be not needed - software can cache the color palette in system memory. This is an array of 256 identical register entries; the register fields below apply to each entry.

Color Palette

Offset: 0x500..0x5ff | Byte Offset: 0x1400..0x17fc | Read/Write: WO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	C_COLOR_PALETTE_B: Blue Color Palette
15:8	X	C_COLOR_PALETTE_G: Green Color Palette
7:0	X	C_COLOR_PALETTE_R: Red Color Palette

25.13.2 DC_WINC_C_PALETTE_COLOR_EXT_0

Window C Palette Color Extension

Palette extension for 1-bpp, 2-bpp, and 4-bpp. These bits provide the upper most significant bits for indexing the color palette.

Offset: 0x600 │ Byte Offset: 0x1800 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:1	X	C_PALETTE_COLOR_EXT: Window C Palette Color Extension. Bits 7:1 are used for 1-bpp mode, bits 7:2 are used for 2-bpp mode, and bits 7:4 are used for 4-bpp mode

25.13.3 DC_WINC_C_H_FILTER_P00_0

Window C Horizontal Filter phase 00

Horizontal scaling filter coefficients.

Horizontal scaling filter is a 6-tap filter with 4-bit positional phase.

- Coefficients 0 and 5 are 3-bit signed values ranging from -4 to 3.
- Coefficients 1 and 4 are 5-bit signed values ranging from -16 to 15.
- Coefficients 2 and 3 are 8-bit unsigned values ranging from 0 to 128.
- Coefficient 0 is the multiplier for the earliest pixel (P0) in the group of 6 pixels and coefficient 5 is the multiplier for the latest pixel (P5) in the group. The output pixel positional phase is defined as centered in P2 if the positional phase is 0 or proportionally in between P2 and P3 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically, and software should never program the sum of all coefficients for a phase to be more than 128.

For each horizontal positional phase, the 6 filter coefficients require 32 register bits. Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Offset: 0x601 │ Byte Offset: 0x1804 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P00C5: Phase 00 coefficient 5 (typically 0)
28:24	X	C_H_FILTER_P00C4: Phase 00 coefficient 4 (typically 0)
23:16	X	C_H_FILTER_P00C3: Phase 00 coefficient 3 (typically 0)
15:8	X	C_H_FILTER_P00C2: Phase 00 coefficient 2 (typically 128)
7:3	X	C_H_FILTER_P00C1: Phase 00 coefficient 1 (typically 0)
2:0	X	C_H_FILTER_P00C0: Phase 00 coefficient 0 (typically 0)

25.13.4 DC_WINC_C_H_FILTER_P01_0

Window C Horizontal Filter phase 01

Offset: 0x602 | Byte Offset: 0x1808 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P01C5: Phase 01 coefficient 5 (typically 1)
28:24	X	C_H_FILTER_P01C4: Phase 01 coefficient 4 (typically -2)
23:16	X	C_H_FILTER_P01C3: Phase 01 coefficient 3 (typically 8)
15:8	X	C_H_FILTER_P01C2: Phase 01 coefficient 2 (typically 124)
7:3	X	C_H_FILTER_P01C1: Phase 01 coefficient 1 (typically -4)
2:0	X	C_H_FILTER_P01C0: Phase 01 coefficient 0 (typically 1)

25.13.5 DC_WINC_C_H_FILTER_P02_0

Window C Horizontal Filter phase 02

Offset: 0x603 | Byte Offset: 0x180c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P02C5: Phase 02 coefficient 5 (typically 1)
28:24	X	C_H_FILTER_P02C4: Phase 02 coefficient 4 (typically -5)
23:16	X	C_H_FILTER_P02C3: Phase 02 coefficient 3 (typically 17)
15:8	X	C_H_FILTER_P02C2: Phase 02 coefficient 2 (typically 122)
7:3	X	C_H_FILTER_P02C1: Phase 02 coefficient 1 (typically -8)
2:0	X	C_H_FILTER_P02C0: Phase 02 coefficient 0 (typically 1)

25.13.6 DC_WINC_C_H_FILTER_P03_0

Window C Horizontal Filter phase 03

Offset: 0x604 | Byte Offset: 0x1810 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P03C5: Phase 03 coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P03C4: Phase 03 coefficient 4 (typically -7)
23:16	X	C_H_FILTER_P03C3: Phase 03 coefficient 3 (typically 27)
15:8	X	C_H_FILTER_P03C2: Phase 03 coefficient 2 (typically 115)
7:3	X	C_H_FILTER_P03C1: Phase 03 coefficient 1 (typically -11)
2:0	X	C_H_FILTER_P03C0: Phase 03 coefficient 0 (typically 2)

25.13.7 DC_WINC_C_H_FILTER_P04_0

Window C Horizontal Filter phase 04

Offset: 0x605 | Byte Offset: 0x1814 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P04C5: Phase 04 coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P04C4: Phase 04 coefficient 4 (typically -9)
23:16	X	C_H_FILTER_P04C3: Phase 04 coefficient 3 (typically 37)
15:8	X	C_H_FILTER_P04C2: Phase 04 coefficient 2 (typically 109)
7:3	X	C_H_FILTER_P04C1: Phase 04 coefficient 1 (typically -13)
2:0	X	C_H_FILTER_P04C0: Phase 04 coefficient 0 (typically 2)

25.13.8 DC_WINC_C_H_FILTER_P05_0

Window C Horizontal Filter phase 05

Offset: 0x606 | Byte Offset: 0x1818 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P05C5: Phase 05 coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P05C4: Phase 05 coefficient 4 (typically -11)
23:16	X	C_H_FILTER_P05C3: Phase 05 coefficient 3 (typically 47)
15:8	X	C_H_FILTER_P05C2: Phase 05 coefficient 2 (typically 102)
7:3	X	C_H_FILTER_P05C1: Phase 05 coefficient 1 (typically -15)
2:0	X	C_H_FILTER_P05C0: Phase 05 coefficient 0 (typically 3)

25.13.9 DC_WINC_C_H_FILTER_P06_0

Window C Horizontal Filter phase 06

Offset: 0x607 | Byte Offset: 0x181c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P06C5: Phase 06 coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P06C4: Phase 06 coefficient 4 (typically -13)
23:16	X	C_H_FILTER_P06C3: Phase 06 coefficient 3 (typically 56)
15:8	X	C_H_FILTER_P06C2: Phase 06 coefficient 2 (typically 94)
7:3	X	C_H_FILTER_P06C1: Phase 06 coefficient 1 (typically -15)
2:0	X	C_H_FILTER_P06C0: Phase 06 coefficient 0 (typically 3)

25.13.10 DC_WINC_C_H_FILTER_P07_0

Window C Horizontal Filter phase 07

Offset: 0x608 | Byte Offset: 0x1820 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P07C5: Phase 07 coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P07C4: Phase 07 coefficient 4 (typically -14)
23:16	X	C_H_FILTER_P07C3: Phase 07 coefficient 3 (typically 67)
15:8	X	C_H_FILTER_P07C2: Phase 07 coefficient 2 (typically 85)
7:3	X	C_H_FILTER_P07C1: Phase 07 coefficient 1 (typically -16)
2:0	X	C_H_FILTER_P07C0: Phase 07 coefficient 0 (typically 3)

25.13.11 DC_WINC_C_H_FILTER_P08_0

Window C Horizontal Filter phase 08

Offset: 0x609 | Byte Offset: 0x1824 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P08C5: Phase 08 coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P08C4: Phase 08 coefficient 4 (typically -15)
23:16	X	C_H_FILTER_P08C3: Phase 08 coefficient 3 (typically 76)
15:8	X	C_H_FILTER_P08C2: Phase 08 coefficient 2 (typically 76)
7:3	X	C_H_FILTER_P08C1: Phase 08 coefficient 1 (typically -15)
2:0	X	C_H_FILTER_P08C0: Phase 08 coefficient 0 (typically 3)

25.13.12 DC_WINC_C_H_FILTER_P09_0

Window C Horizontal Filter phase 09

Offset: 0x60a | Byte Offset: 0x1828 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P09C5: Phase 09 coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P09C4: Phase 09 coefficient 4 (typically -16)
23:16	X	C_H_FILTER_P09C3: Phase 09 coefficient 3 (typically 85)
15:8	X	C_H_FILTER_P09C2: Phase 09 coefficient 2 (typically 67)
7:3	X	C_H_FILTER_P09C1: Phase 09 coefficient 1 (typically -14)
2:0	X	C_H_FILTER_P09C0: Phase 09 coefficient 0 (typically 3)

25.13.13 DC_WINC_C_H_FILTER_P0A_0

Window C Horizontal Filter phase 0A

Offset: 0x60b │ Byte Offset: 0x182c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P0AC5: Phase 0A coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P0AC4: Phase 0A coefficient 4 (typically -15)
23:16	X	C_H_FILTER_P0AC3: Phase 0A coefficient 3 (typically 94)
15:8	X	C_H_FILTER_P0AC2: Phase 0A coefficient 2 (typically 56)
7:3	X	C_H_FILTER_P0AC1: Phase 0A coefficient 1 (typically -13)
2:0	X	C_H_FILTER_P0AC0: Phase 0A coefficient 0 (typically 3)

25.13.14 DC_WINC_C_H_FILTER_P0B_0

Window C Horizontal Filter phase 0B

Offset: 0x60c │ Byte Offset: 0x1830 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P0BC5: Phase 0B coefficient 5 (typically 3)
28:24	X	C_H_FILTER_P0BC4: Phase 0B coefficient 4 (typically -15)
23:16	X	C_H_FILTER_P0BC3: Phase 0B coefficient 3 (typically 102)
15:8	X	C_H_FILTER_P0BC2: Phase 0B coefficient 2 (typically 47)
7:3	X	C_H_FILTER_P0BC1: Phase 0B coefficient 1 (typically -11)
2:0	X	C_H_FILTER_P0BC0: Phase 0B coefficient 0 (typically 2)

25.13.15 DC_WINC_C_H_FILTER_P0C_0

Window C Horizontal Filter phase 0C

Offset: 0x60d │ Byte Offset: 0x1834 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P0CC5: Phase 0C coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P0CC4: Phase 0C coefficient 4 (typically -13)
23:16	X	C_H_FILTER_P0CC3: Phase 0C coefficient 3 (typically 109)
15:8	X	C_H_FILTER_P0CC2: Phase 0C coefficient 2 (typically 37)
7:3	X	C_H_FILTER_P0CC1: Phase 0C coefficient 1 (typically -9)
2:0	X	C_H_FILTER_P0CC0: Phase 0C coefficient 0 (typically 2)

25.13.16 DC_WINC_C_H_FILTER_P0D_0

Window C Horizontal Filter phase 0D

Offset: 0x60e │ Byte Offset: 0x1838 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P0DC5: Phase 0D coefficient 5 (typically 2)
28:24	X	C_H_FILTER_P0DC4: Phase 0D coefficient 4 (typically -11)
23:16	X	C_H_FILTER_P0DC3: Phase 0D coefficient 3 (typically 115)
15:8	X	C_H_FILTER_P0DC2: Phase 0D coefficient 2 (typically 27)
7:3	X	C_H_FILTER_P0DC1: Phase 0D coefficient 1 (typically -7)
2:0	X	C_H_FILTER_P0DC0: Phase 0D coefficient 0 (typically 2)

25.13.17 DC_WINC_C_H_FILTER_P0E_0

Window C Horizontal Filter phase 0E

Offset: 0x60f │ Byte Offset: 0x183c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P0EC5: Phase 0E coefficient 5 (typically 1)
28:24	X	C_H_FILTER_P0EC4: Phase 0E coefficient 4 (typically -8)
23:16	X	C_H_FILTER_P0EC3: Phase 0E coefficient 3 (typically 122)
15:8	X	C_H_FILTER_P0EC2: Phase 0E coefficient 2 (typically 17)
7:3	X	C_H_FILTER_P0EC1: Phase 0E coefficient 1 (typically -5)
2:0	X	C_H_FILTER_P0EC0: Phase 0E coefficient 0 (typically 1)

25.13.18 DC_WINC_C_H_FILTER_P0F_0

Window C Horizontal Filter phase 0F

Offset: 0x610 │ Byte Offset: 0x1840 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	C_H_FILTER_P0FC5: Phase 0F coefficient 5 (typically 1)
28:24	X	C_H_FILTER_P0FC4: Phase 0F coefficient 4 (typically -4)
23:16	X	C_H_FILTER_P0FC3: Phase 0F coefficient 3 (typically 124)
15:8	X	C_H_FILTER_P0FC2: Phase 0F coefficient 2 (typically 8)
7:3	X	C_H_FILTER_P0FC1: Phase 0F coefficient 1 (typically -2)
2:0	X	C_H_FILTER_P0FC0: Phase 0F coefficient 0 (typically 1)

25.13.19 DC_WINC_C_CSC_YOF_0

Window C CSC Y Offset

Color Space Conversion coefficients

The CSC can be used for YUV to RGB conversion with brightness and hue/saturation control. The CSC can only be enabled for window C controlled by CSC_ENABLE register bits.

For Y color, the Y offset is applied first and saturation (clipping) is performed immediately after the Y offset is applied.

- R = sat(KYRGB * sat(Y + YOF) + KUR * U + KVR * V)
- G = sat(KYRGB * sat(Y + YOF) + KUG * U + KVG * V)
- B = sat(KYRGB * sat(Y + YOF) + KUB * U + KVB * V)

Saturation and rounding is performed in the range of 0 to 255 for the above equations.

Typical values are:

- YOF = -16.000, KYRGB = 1.1644
- KUR = 0.0000, KVR = 1.5960
- KUG = -0.3918, KVG = -0.8130
- KUB = 2.0172, KVB = 0.0000

KUR and KVB are typically 0.0000 but they may be programmed non-zero for hue rotation.

The CSC can also take RGB input, in which case YOF, KVB, KUG, and KUR should be programmed to 0, and KYRGB will be forced to 0 by the hardware for generating R and B. KYRGB will not be forced to 0 for generating G. KVR, KYRGB, and KUB can be programmed to 1.0 or used as gain control for R, G, B, correspondingly. Note that color value ranges from 0 to 255 for Y, R, G, B, and -128 to 127 for U and V.

Offset: 0x611 │ Byte Offset: 0x1844 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_CSC_YOF: Y Offset in s.7.0 format

25.13.20 DC_WINC_C_CSC_KYRGB_0

Window C CSC Y Coefficient (gain) for RGB

Offset: 0x612 │ Byte Offset: 0x1848 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	C_CSC_KYRGB: Y Gain for R, G, B colors in 2.8 format

25.13.21 DC_WINC_C_CSC_KUR_0

Window C CSC U coefficient for R

Offset: 0x613 │ Byte Offset: 0x184c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	C_CSC_KUR: U coefficients for R in s.2.8 format

25.13.22 DC_WINC_C_CSC_KVR_0

Window C CSC V coefficient for R

Offset: 0x614 | Byte Offset: 0x1850 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	C_CSC_KVR: V coefficients for R in s.2.8 format

25.13.23 DC_WINC_C_CSC_KUG_0

Window C CSC U coefficient for G

Offset: 0x615 | Byte Offset: 0x1854 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	C_CSC_KUG: U coefficients for G in s.1.8 format

25.13.24 DC_WINC_C_CSC_KVG_0

Window C CSC V coefficient for G

Offset: 0x616 | Byte Offset: 0x1858 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9:0	X	C_CSC_KVG: V coefficients for G in s.1.8 format

25.13.25 DC_WINC_C_CSC_KUB_0

Window C CSC U coefficient for B

Offset: 0x617 | Byte Offset: 0x185c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	C_CSC_KUB: U coefficients for B in s.2.8 format

25.13.26 DC_WINC_C_CSC_KVB_0

Window C CSC V coefficient for B

Offset: 0x618 | Byte Offset: 0x1860 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10:0	X	C_CSC_KVB: V coefficients for B in s.2.8 format

25.13.27 DC_WINC_C_V_FILTER_P00_0

Window C Vertical Filter phase 00

Vertical scaling filter coefficients

Vertical scaling filter is a 2-tap filter with 4-bit positional phase. Coefficients 0 and 1 are 8-bit unsigned values ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2-pixels and coefficient 1 is the multiplier for the later pixel (P1) in the group. The output pixel positional phase is defined as centered in P0 if the positional phase is 0 or proportionally in between P0 and P1 if the positional phase is larger than 0.

Sum of all coefficients for each phase should be 128 typically; therefore coefficient 1 can be calculated from (1 - coefficient 0) and only coefficient 0 is programmed.

For each vertical positional phase, the filter coefficient requires 8 register bits. Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Offset: 0x619 │ Byte Offset: 0x1864 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P00C0: Phase 00 coefficient 0 (typically 128)

25.13.28 DC_WINC_C_V_FILTER_P01_0

Window C Vertical Filter phase 01

Offset: 0x61a │ Byte Offset: 0x1868 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P01C0: Phase 01 coefficient 0 (typically 120)

25.13.29 DC_WINC_C_V_FILTER_P02_0

Window C Vertical Filter phase 02

Offset: 0x61b │ Byte Offset: 0x186c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P02C0: Phase 02 coefficient 0 (typically 112)

25.13.30 DC_WINC_C_V_FILTER_P03_0

Window C Vertical Filter phase 03

Offset: 0x61c │ Byte Offset: 0x1870 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P03C0: Phase 03 coefficient 0 (typically 104)

25.13.31 DC_WINC_C_V_FILTER_P04_0

Window C Vertical Filter phase 04

Offset: 0x61d | Byte Offset: 0x1874 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P04C0: Phase 04 coefficient 0 (typically 96)

25.13.32 DC_WINC_C_V_FILTER_P05_0

Window C Vertical Filter phase 05

Offset: 0x61e | Byte Offset: 0x1878 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P05C0: Phase 05 coefficient 0 (typically 88)

25.13.33 DC_WINC_C_V_FILTER_P06_0

Window C Vertical Filter phase 06

Offset: 0x61f | Byte Offset: 0x187c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P06C0: Phase 06 coefficient 0 (typically 80)

25.13.34 DC_WINC_C_V_FILTER_P07_0

Window C Vertical Filter phase 07

Offset: 0x620 | Byte Offset: 0x1880 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P07C0: Phase 07 coefficient 0 (typically 72)

25.13.35 DC_WINC_C_V_FILTER_P08_0

Window C Vertical Filter phase 08

Offset: 0x621 | Byte Offset: 0x1884 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P08C0: Phase 08 coefficient 0 (typically 64)

25.13.36 DC_WINC_C_V_FILTER_P09_0

Window C Vertical Filter phase 09

Offset: 0x622 | Byte Offset: 0x1888 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P09C0: Phase 09 coefficient 0 (typically 56)

25.13.37 DC_WINC_C_V_FILTER_P0A_0

Window C Vertical Filter phase 0A

Offset: 0x623 | Byte Offset: 0x188c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P0AC0: Phase 0A coefficient 0 (typically 48)

25.13.38 DC_WINC_C_V_FILTER_P0B_0

Window C Vertical Filter phase 0B

Offset: 0x624 | Byte Offset: 0x1890 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P0BC0: Phase 0B coefficient 0 (typically 40)

25.13.39 DC_WINC_C_V_FILTER_P0C_0

Window C Vertical Filter phase 0C

Offset: 0x625 | Byte Offset: 0x1894 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P0CC0: Phase 0C coefficient 0 (typically 32)

25.13.40 DC_WINC_C_V_FILTER_P0D_0

Window C Vertical Filter phase 0D

Offset: 0x626 | Byte Offset: 0x1898 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P0DC0: Phase 0D coefficient 0 (typically 24)

25.13.41 DC_WINC_C_V_FILTER_P0E_0

Window C Vertical Filter phase 0E

Offset: 0x627 | Byte Offset: 0x189c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P0EC0: Phase 0E coefficient 0 (typically 16)

25.13.42 DC_WINC_C_V_FILTER_P0F_0

Window C Vertical Filter phase 0F

Offset: 0x628 │ Byte Offset: 0x18a0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	C_V_FILTER_P0FC0: Phase 0F coefficient 0 (typically 8)

The registers under DC_WIN are double buffered.

25.13.43 DC_WIN_C_WIN_OPTIONS_0

Window C Options

Class: Display Window Settings

Display Window C parameters

Offset: 0x700 │ Byte Offset: 0x1c00 │ Read/Write: R/W │ Reset: 0x00000000 (0bx0xxxxxxxxxxxx0x0xxxxxxxx0xxxx)

Bit	Reset	Description
30	0x0	C_WIN_ENABLE: Window C Window enable 0 = DISABLE 1 = ENABLE
22	X	C_YUV_RANGE_EXPAND: Window C Enable range expansion in the cases where RANGEREDFRM is 1 from mpd. Formula: Y = clip((Y-128)*2 + 128); Cb = clip((Cb-128)*2 + 128); Cr = clip((Cr-128)*2 + 128); where clip() function clips between 0 and 255. 0 = DISABLE 1 = ENABLE
20	X	C_DV_ENABLE: Window C Digital Vibrance Enable 0 = DISABLE 1 = ENABLE
18	X	C_CSC_ENABLE: Window C Color Space Conversion Enable. This controls the color space conversion and should be enabled for YCbCr/YUV color modes for conversion to B8G8R8 and for hue and saturation control. This can also be used for gain control for RGB color modes. 0 = DISABLE 1 = ENABLE
16	0x0	C_CP_ENABLE: Window C Color Palette Enable. This controls the color palette and should be enabled for palletized color modes. For non-palletized color modes, the color palette can be enabled for gamma correction. 0 = DISABLE 1 = ENABLE
14	0x0	C_V_FILTER_UV_ALIGN: Window C V Filter UV Alignment. This is effective only when vertical scaling filter is enabled and only on these formats YCbCr420P, YUV420P, YCbCr422R, YUV422R, YCbCr422RA YUV422RA. When UV alignment is enabled, the chroma components are aligned to the even number of luma component lines. When disabled the chroma components are aligned to half a pixel below the corresponding even number of luma component lines. It is usually disabled unless the incoming video stream specifically indicates otherwise. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	X	C_V_FILTER_OPTIMIZE: Window C V Filter Optimization. This is effective only when vertical scaling filter is enabled. This can be used (enabled) to temporarily disable the vertical scaling filter when the vertical scaling DDA fraction is zero. In this case the next line is not fetched from memory to save bandwidth and power. This feature cannot be used in 420P/422R/422RA formats. 0 = DISABLE 1 = ENABLE
10	X	C_V_FILTER_ENABLE: Window C V Filter Enable. This controls the V scaling filter and is effective only for non-palletized color modes. If V filter is disabled, only one line is read from memory for each output line. NOTE: This feature is not supported. This bit is ignored and is a don't care. 0 = DISABLE 1 = ENABLE
8	X	C_H_FILTER_ENABLE: Window C H Filter Enable. This controls H scaling filter and is effective only for non-palletized color modes. 0 = DISABLE 1 = ENABLE
6	X	C_COLOR_EXPAND: Window C 12/15/16/18-to-24 bpp color expansion. This bit should be enabled only for 12-bpp B4G4R4A4, 15-bpp B5G5R5A, 16-bpp B5G6R5, 18-bpp B6G6R6 color modes. If enabled, the color conversion is performed prior to horizontal scaling. 0 = DISABLE 1 = ENABLE
4	0x0	C_SCAN_COLUMN: Window C Scanning direction. 0= Scan in horizontal direction (for 0 or 180 degrees). 1= Scan in vertical direction (for 90 or 270 degrees) 0 = DISABLE 1 = ENABLE
2	X	C_V_DIRECTION: Window C Vertical (Y) drawing Direction 0 = INCREMENT 1 = DECREMENT
0	X	C_H_DIRECTION: Window C Horizontal (X) drawing Direction 0 = INCREMENT 1 = DECREMENT

25.13.44 DC_WIN_C_BYTE_SWAP_0

Window C Byte Swap

Offset: 0x701 │ Byte Offset: 0x1c04 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2:0	X	C_BYTE_SWAP: Window C Byte Swap. This controls byte swap of frame data read from memory prior to any data processing in the display module. 000= no byte swap (3 2 1 0) 001= byte swap for each 2-byte word (2 3 0 1) 010= byte swap for each 4-byte word (0 1 2 3) 011= word swap for each 4-byte word (1 0 3 2) 100= byte0 swapped with byte2 (3 0 1 2) 101= left shift every byte (2 1 0 3) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW 4 = SWAP02 5 = SWAPLEFT

25.13.45 DC_WIN_C_COLOR_DEPTH_0

Window C Color Depth. For YCbCr data format, Cb and Cr are 8-bit unsigned values. For YUV data format, U and V are 8-bit signed values. YCbCr422R is similar to YCbCr422P but the Cb and Cr are shared vertically.

YUV422R is similar to YUV422P, but the U and V are shared vertically. YCbCr422RA is the same as YCbCr422R in memory and YUV422RA is the same as YUV422R in memory but while reading from memory, for YCbCr422RA and YUV422RA, chroma averaging is applied for each pixel pair so that they can be processed as YUV422 by the display pipeline.

For YCbCr422R and YUV422R, every other chroma pixels are not used (discarded) by the display pipeline. B6x2G6x2R6x2A8 is similar to B8G8R6A8 but with the 2 LSBs zeroed out. R6x2G6x2B6x2A8 is similar to R8G8B6A8 but with the 2 LSBs zeroed out.

Some formats have NVCF_ or T_ prefix aliases for NVidia surface format naming convention, wherein left-to-right component names correspond to MSB to LSB bits within a word. In old names without NVCF_ or T_, left-to-right component names generally map LSB to MSB within a word.

Offset: 0x703 │ Byte Offset: 0x1c0c │ Read/Write: R/W │ Reset: 0x0000000c (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx001100)

Bit	Reset	Description
5:0	B8G8R8A8	C_COLOR_DEPTH: Window C Color Depth. Supported color depths are: P8 = 8-bpp (palletized) B4G4R4A4 = 12-bpp B4G4R4 B5G5R5A = 15-bpp B5G5R5 AB5G5R5 = 15-bpp B5G5R5 B5G6R5 = 16-bpp B5G6R5 B8G8R8A8 = 32-bpp B8G8R8A8 R8G8B8A8 = 32-bpp R8G8B8A8 YCbCr422 = 16-bpp YCbCr422 packed YUV422 = 16-bpp YUV422 YCbCr420P = 16-bpp YCbCr420 planar YUV420P = 16-bpp YUV420 planar YCbCr422P = 16-bpp YCbCr422 planar YUV422P = 16-bpp YUV422 planar YCbCr422R = 16-bpp YCbCr422 rotated planar YUV422R = 16-bpp YUV422 rotated planar YCbCr422RA= 16-bpp YCbCr422 rotated planar with chroma averaging YUV422RA = 16-bpp YUV422 rotated planar with chroma averaging Reserved values: 0,1,2,14,15,24,25. 3 = T_P8, P8 4 = T_A4R4G4B4, B4G4R4A4 5 = T_A1R5G5B5, B5G5R5A 6 = T_R5G6B5, B5G6R5 7 = T_R5G5B5A1, AB5G5R5 12 = T_A8R8G8B8, B8G8R8A8 13 = T_A8B8G8R8, R8G8B8A8 16 = T_U8_Y8__V8_Y8, CbYCrY422, YCbCr422 17 = T_U8_Y8__V8_Y8_TRUE, UYVY422, YUV422, TRUE_U8_Y8__V8_Y8 18 = T_Y8___U8___V8_N420, YCbCr420P 19 = T_Y8___U8___V8_N420_TRUE, YUV420P, TRUE_Y8___U8___V8_N420 20 = T_Y8___U8___V8_N422, YCbCr422P 21 = T_Y8___U8___V8_N422_TRUE, YUV422P, TRUE_Y8___U8___V8_N422 22 = T_Y8___U8___V8_N422R, YCbCr422RP, YCbCr422R 23 = T_Y8___U8___V8_N422R_TRUE, YUV422RP, YUV422R, TRUE_Y8___U8___V8_N422R (new format, reserved for future use) 26 = T_Y8, Y8 27 = T_A4B4G4R4, R4G4B4A4 28 = T_A1B5G5R5, R5G5B5A 29 = T_B5G5R5A1, AR5G5B5 30 = T_X1R5G5B5, B5G5R5X1 31 = T_R5G5B5X1, X1B5G5R5 32 = T_X1B5G5R5, R5G5B5X1 33 = T_B5G5R5X1, X1R5G5B5 34 = T_B5G6R5, R5G6B5 35 = T_B8G8R8A8 36 = T_R8G8B8A8

Bit	Reset	Description
		37 = T_X8R8G8B8, B8G8R8X8 38 = T_X8B8G8R8, R8G8B8X8 39 = T_A8Y8U8V8 40 = T_V8U8Y8A8 41 = T_Y8___U8___V8_N444, YCbCr444P 42 = T_Y8___U8V8_N420, YCrCb420SP 43 = T_Y8___V8U8_N420, YCbCr420SP 44 = T_Y8___U8V8_N422, YCrCb422SP 45 = T_Y8___V8U8_N422, YCbCr422SP 46 = T_Y8___U8V8_N422R, YCrCb422RSP 47 = T_Y8___V8U8_N422R, YCbCr422RSP 48 = T_Y8___U8V8_N444, YCrCb444SP 49 = T_Y8___V8U8_N444, YCbCr444SP 50 = T_A8Y8U8V8_TRUE, TRUE_A8Y8U8V8 51 = T_V8U8Y8A8_TRUE, TRUE_V8U8Y8A8 52 = T_Y8___U8___V8_N444_TRUE, YUV444P, TRUE_Y8___U8___V8_N444 53 = T_Y8___U8V8_N420_TRUE, YVU420SP, TRUE_Y8___U8V8_N420 54 = T_Y8___V8U8_N420_TRUE, YUV420SP, TRUE_Y8___V8U8_N420 55 = T_Y8___U8V8_N422_TRUE, YVU422SP, TRUE_Y8___U8V8_N422 56 = T_Y8___V8U8_N422_TRUE, YUV422SP, TRUE_Y8___V8U8_N422 57 = T_Y8___U8V8_N422R_TRUE, YVU422RSP, TRUE_Y8___U8V8_N422R 58 = T_Y8___V8U8_N422R_TRUE, YUV422RSP, TRUE_Y8___V8U8_N422R 59 = T_Y8___U8V8_N444_TRUE, YUV444SP, TRUE_Y8___U8V8_N444 60 = T_Y8___V8U8_N444_TRUE, YVU444SP,TRUE_Y8___V8U8_N444

25.13.46 DC_WIN_C_POSITION_0

Window C Position

This register defines H position and size of Window C after scaling (if there is any)

Offset: 0x704 │ Byte Offset: 0x1c10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	C_V_POSITION: Window C V Position. This is specified with respect to the top edge of active display area.
12:0	X	C_H_POSITION: Window C H Position. This is specified with respect to the left edge of active display area.

25.13.47 DC_WIN_C_SIZE_0

Window C Size

This register defines V position and size of Window C after scaling (if there is any)

> **Note:** Programming on window size should guarantee the whole window is inside the active area, otherwise extra rows/columns will be fetched and discarded which affects performance.

Offset: 0x705 │ Byte Offset: 0x1c14 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	C_V_SIZE: Window C V Size (lines). This is the vertical size after scaling.
12:0	X	C_H_SIZE: Window C H Size (pixels). This is the horizontal size after scaling.

25.13.48 DC_WIN_C_PRESCALED_SIZE_0

Window C Pre-scaled Size

This register defines Window C pre-scaled size.

The H pre-scaled size is needed to determine how many bytes to fetch from memory per line and this parameter must be programmed exactly as needed taking into account the scaling factor. For planar YUV or YCbCr data formats, this parameter refer to the H pre-scaled of the Y plane.

The total number of lines to be fetched from memory is determined by post-scale V size but V pre-scaled size is needed to 'clamp' the last valid line if the vertical DDA is exactly or slightly beyond the specified V pre-scaled size.

H pre-scaled size ideally should be in terms of pixel but then hardware needs to convert this precisely to bytes to determine the amount of data to request from memory.

Offset: 0x706 │ Byte Offset: 0x1c18 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	C_V_PRESCALED_SIZE: Window C V Pre-scaled Size (lines). In 420P/422R/422RA formats, it must be even.
14:0	X	C_H_PRESCALED_SIZE: Window C H Pre-scaled Size (bytes). In 420P and 422P formats, it must be even.

25.13.49 DC_WIN_C_H_INITIAL_DDA_0

Window C H Initial DDA

The first pixel of the pre-scaled image is always used to output the first pixel, so essentially this is the same as forcing the H Initial DDA integer portion to 1 initially even though user typically programs this to 0. If it makes the implementation easier, it is possible to force software to program the Initial DDA integer portion to 1. Similarly with the V Initial DDA.

Offset: 0x707 │ Byte Offset: 0x1c1c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	C_H_INITIAL_DDA: Window C H Initial DDA (4.12). This is typically programmed to 0.0

25.13.50 DC_WIN_C_V_INITIAL_DDA_0

Window C V Initial DDA

Offset: 0x708 │ Byte Offset: 0x1c20 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	C_V_INITIAL_DDA: Window C V Initial DDA (4.12). This is typically programmed to 0.0 for both non-interlaced and interlaced sources.

25.13.51 DC_WIN_C_DDA_INCREMENT_0

Window C DDA Increment

DDA increment is typically calculated by dividing (Pre-scaled size in pixels - 1) by (Post-scaled size in pixels - 1). The result should be rounded up and expressed as 4.12 format (4-bit integer and 12-bit fraction). For non-filtered image this value can be

slightly larger so that it is not missing the last row/column. Reference programming values (H and V should be calculated separately depending on filter on/off and sizes):

- Filter on: min(round((prescaled_size_in_pixels - 1) * 0x1000 /(post_scaled_size_in_pixels - 1)), MAX)
- Filter off: min(round(prescaled_size_in_pixels * 0x1000 / post_scaled_size_in_pixels - 1) - 0.5), MAX)

Where the value of MAX is as follows:

- For V_DDA_INCREMENT: 15.0 (0xF000)
- For H_DDA_INCREMENT: 4.0 (0x4000) for 4 Bytes/pix formats.
  8.0 (0x8000) for 2 Bytes/pix formats.

They are theoretically the biggest values that guarantee not displaying beyond an image boundary. If the DDA increment is less than 1.0 then image will be upscaled and if DDA increment is more than 1.0, then the image will be downscaled.

Offset: 0x709 │ Byte Offset: 0x1c24 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	C_V_DDA_INCREMENT: Window C Vertical DDA Increment (4.12). This should be set to 1.0 if there is no scaling. Maximum value is 15.0 regardless of the number of bytes per pixel.
15:0	X	C_H_DDA_INCREMENT: Window C Horizontal DDA Increment (4.12). This should be set to 1.0 if there is no scaling. The maximum value for downscaling depends on the number of bytes per pixel. For 4-byte/pixel modes (32-bpp) the maximum value is 4.0 and for all other modes the maximum value is 8.0.

25.13.52 DC_WIN_C_LINE_STRIDE_0

Window C Line Stride

Offset: 0x70a │ Byte Offset: 0x1c28 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	C_UV_LINE_STRIDE: Window C Line Stride for Chroma. This is stride (in bytes) for planar YUV or YCbCr data formats for the chroma plane, with the restriction that it must be programmed to be multiples of 4 (16 if tiled or in horizontal flipping) This is not used (ignored) for other non-planar data formats.
15:0	X	C_LINE_STRIDE: Window C Line Stride. This is stride (in bytes) for all non-planar data formats. If the memory surface is tiled, the stride needs to be a multiple of 16. If H_DIRECTION of window C is set to DECREMENT, the stride also needs to be a multiple of 16. For planar YUV or YCbCr data formats, this is stride (in bytes) for the luma plane with the restriction that it must be multiples of 8 (16 if tiled or in horizontal flipping) For tiled surface this value may affect starting address of a window. Refer to the comment of START_ADDR for detail.

25.13.53 DC_WIN_C_BUFFER_ADDR_MODE_0

Memory Controller Tiling definitions

Offset: 0x70d │ Byte Offset: 0x1c34 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxxxxxxxxxx0)

Bit	Reset	Description
16	0x0	C_UV_TILE_MODE: Window C Memory surface tiling mode. For YUV planar pixel format, this specifies tiling mode for the U/V plane. 0 = LINEAR 1 = TILED

Bit	Reset	Description
0	0x0	C_TILE_MODE: Window C Memory surface tiling mode. For YUV planar pixel format, this specifies tiling mode for the Y plane. 0 = LINEAR 1 = TILED

25.13.54 DC_WIN_C_DV_CONTROL_0

Window C Digital Vibrance Control

If enabled, Digital Vibrance is applied after H and V scaling and after color palette or color space conversion logic but before color keying multiplexer and before cursor multiplexer.

- After DV, new R = R + (2R - G - B) * FR, where FR is fraction from 0 to 7/8
- After DV, new G = G + (2G - R - B) * FG, where FG is fraction from 0 to 7/8
- After DV, new B = B + (2B - R - G) * FB, where FB is fraction from 0 to 7/8

Offset: 0x70e │ Byte Offset: 0x1c38 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18:16	X	C_DV_CONTROL_B: Digital Vibrance control for B
10:8	X	C_DV_CONTROL_G: Digital Vibrance control for G
2:0	X	C_DV_CONTROL_R: Digital Vibrance control for R

25.13.55 DC_WIN_C_BLEND_NOKEY_0

Blend Control for this window area where color key is enabled but the pixel color is not within the color key range (color key not match). This is valid for all overlapping condition but only if there is no overlap with other window with higher priority color key enabled and color key not match.

Class: Display Color Keying and Blending

Color keying and blending of the display windows are done prior to cursor blending. Cursor always goes on top of the blended windows. Blending is controlled independently on each possible overlap area of the display windows. If 3 windows are enabled there are 7 possible overlap area combinations. For every window in each overlap area combination, color key can be disabled or enabled. Also, for every window in each overlap area combination, there is a corresponding window blend control parameter and a window blend weight parameter. The window blend control parameter is always effective but the window blend weight is not always used. The window blend weight can also be derived from pixel alpha value or from the reverse of other overlapping windows weight.

Color keying has the highest priority for display window blending. Color key consists of a range of color which is searched independently for each window. If more than 1 window color key is enabled, then Window A color key has the highest priority, followed by Window B color key, and then followed by Window C color key. Two sets of color key ranges (Color Key 0 and Color Key 1) can be defined; they are shared for all windows. It is possible to use both color key sets for the same window or for two separate windows.

The two sets of color key ranges should not overlap. If they do, the overlapped colors are treated as if they are part of Color Key 0 and not part of Color Key 1.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for any overlap condition and the window pixel is not within the color key range (key not match), then the window pixel

will not be blended with other overlapping window pixels but it will be weighted. The weight is not dependent on the overlap condition it is controlled by the same set of parameters for all overlapping condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is enabled for a particular overlap condition and the window pixel is within the color key range (key match), then the window pixel will be weighted and blended with other overlapping pixels and this is controlled separately for each overlap condition.

Assuming that there is no overlap with other higher priority window with color key not matched, if a window color key is disabled then the window pixel will be blended with other overlapping pixels and this is controlled separately for each overlap condition.

Display Color Key Parameters

For B4G4R4A4, B5G6R5A, and B5G6R5 modes, the color key should be compared prior to the color conversion to 24-bpp and the unused least significant bits of the pixel are filled with zeros.

For palletized mode, the color key is compared prior to the color palette, and the palletized color is compared against the green color key values/mask.

For YUV mode, U and V are offset by +128 before performing the color key comparison.

In all cases, the color key is compared prior to the horizontal/vertical scaling filter and prior to digital vibrance control.

Both upper and lower values are inclusive.

Offset: 0x70f │ Byte Offset: 0x1c3c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	C_BLEND_WEIGHT1_NOKEY: Window blend weight 1 for color key not match areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	C_BLEND_WEIGHT0_NOKEY: Window blend weight 0 for color key not match areas. For alpha weight, this is used for 1-bit alpha with value of 0.
0	X	C_BLEND_CONTROL_NOKEY: Window blend control for color key not match areas. 0 = Fix weight using window blend weight 0 for color key not matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT

25.13.56 DC_WIN_C_BLEND_1WIN_0

Blend Control for this window area where it does not overlap with other windows.

Offset: 0x710 │ Byte Offset: 0x1c40 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	C_BLEND_WEIGHT1_1WIN: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	C_BLEND_WEIGHT0_1WIN: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
2	X	C_BLEND_CONTROL_1WIN: Window blend control in area where it does not overlap with other windows and either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched.

Bit	Reset	Description
		1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT
1:0	X	C_CKEY_ENABLE_1WIN: Window color key enable. 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

25.13.57 DC_WIN_C_BLEND_2WIN_A_0

Blend Control for this window area that overlaps with window A only.

Offset: 0x711 │ Byte Offset: 0x1c44 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	C_BLEND_WEIGHT1_2WIN_A: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	C_BLEND_WEIGHT0_2WIN_A: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	C_BLEND_CONTROL_2WIN_A: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 3 = Premultipled weight; in this case, the window blend weight is 1. But other windows that use DEPENDENT_WEIGHT get (1 - alpha value),like ALPHA_WEIGHT. This is only valid if the window color format includes an alpha value. This allows a blending equation like: result = win_b * 1 + (1 - win_b_alpha) * win_c 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT 3 = PREMULT_WEIGHT
1:0	X	C_CKEY_ENABLE_2WIN_A: Window color key enable. 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

25.13.58 DC_WIN_C_BLEND_2WIN_B_0

Blend Control for this window area that overlaps with window B only.

Offset: 0x712 │ Byte Offset: 0x1c48 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	C_BLEND_WEIGHT1_2WIN_B: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	C_BLEND_WEIGHT0_2WIN_B: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.
3:2	X	C_BLEND_CONTROL_2WIN_B: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 3 = Premultipled weight; in this case, the window blend weight is 1. But other windows that use DEPENDENT_WEIGHT get (1 - alpha value), like ALPHA_WEIGHT. This is only valid if the window color format includes an alpha value. This allows a blending equation like: result = win_a * 1 + (1 - win_a_alpha) * win_c. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT 3 = PREMULT_WEIGHT
1:0	X	C_CKEY_ENABLE_2WIN_B: Window color key enable. 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

25.13.59 DC_WIN_C_BLEND_3WIN_AB_0

Blend Control for this window area that overlaps with windows A & B only.

Offset: 0x713 │ Byte Offset: 0x1c4c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
23:16	X	C_BLEND_WEIGHT1_3WIN_AB: Window blend weight 1 for color key disabled or color key enabled with key matched areas. This is used only for alpha weight with 1-bit alpha and alpha value of 1.
15:8	X	C_BLEND_WEIGHT0_3WIN_AB: Window blend weight 0 for color key disabled or color key enabled with key matched areas. For alpha weight, this is used for 1-bit alpha with value of 0.

Bit	Reset	Description
3:2	X	C_BLEND_CONTROL_3WIN_AB: Window blend control in area where either color key disabled or color key enabled with key matched. 0 = Fix weight using the window blend weight 0 if color key disabled or color key 0 enabled with key matched, or using the window blend weight 1 if color key 1 enabled with key matched. 1 = Alpha weight using the alpha value. This is only valid if the window color format includes an alpha value. For 1-bit alpha value, if the alpha is 0 window blend weight 0 is used and if alpha is 1, window blend weight 1 is used. 2 = Dependent weight, in this case, the window blend weight is 1 - the weights of the other window that overlaps with this window. Only 1 of the overlapping windows may have this setting. 3 = Premultipled weight; in this case, the window blend weight is 1. But other windows that use DEPENDENT_WEIGHT get (1 - alpha value), like ALPHA_WEIGHT. This is only valid if the window color format includes an alpha value 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_WEIGHT 3 = PREMULT_WEIGHT
1:0	X	C_CKEY_ENABLE_3WIN_AB: Window color key enable. 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable 2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

25.13.60 DC_WIN_C_GLOBAL_ALPHA_0

Window C Global Alpha

When enabled, per pixel alpha values are scaled by GLOBAL_ALPHA before blending: a' = a * GLOBAL_ALPHA/255. When per pixel alpha is not used in blending, global alpha has no affect.

Note: 1 and 4 bits per pixel alpha are quantized back to 1 and 4 bits, respectively, after scaling. For the 1-bit alpha case, finer scaling can be achieved by changing BLEND_WEIGHT* instead.

Offset: 0x715 │ Byte Offset: 0x1c54 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxxxxxxxxxxxx)

Bit	Reset	Description
16	DISABLE	C_GLOBAL_ALPHA_ENABLE: When enabled, color palette *must* also be enabled and programmed. 0 = DISABLE 1 = ENABLE
7:0	X	C_GLOBAL_ALPHA: Global alpha: 0..255 map to 0.0 – 1.0. i.e., 255 yields no-operation. For power, better to set GLOBAL_ALPHA_ENABLE=DISABLE when GLOBAL_ALPHA=255.

25.14 WINBUF_C Registers

The registers under DC_WINBUF are triple-buffered.

25.14.1 DC_WINBUF_C_START_ADDR_0

Window C Start Address

Overview

START_ADDR, BUF_STRIDE, LINE_STRIDE, ADDR_H_OFFSET, ADDR_V_OFFSET combined, specify the starting address of a window buffer in the memory surface. Generally START_ADDR is programmed with the starting address of the memory surface. H/V_OFFSET specifies the address offsets of the pixel at the beginning of the window, with respect to the starting address of the memory surface. (There is an exception to that when memory surface is not well aligned, see “For linear address mode” under Programming Restrictions below.)

Note that "beginning of the window" here means the top-left corner of a window in normal mode, top-right corner in horizontal flipping, bottom-left corner in vertical flipping, and bottom-right in horizontal+vertical flipping.

Starting Address Calculation

These formulae are same for both tiled and linear modes. However, for linear mode, the addresses calculated are real physical addresses. For tiled mode, these addresses will be translated to the real physical addresses by the memory controller client before being used.

- When a window is host triggered, the starting address of a window is calculated as follows by hardware.
  - non-YUV-planar modes

    starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar modes

    Y-starting-address = START_ADDR + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2 where denom1/denom2 equal to 1 or 2 depending on the actual planar format, derived natively by hardware.
- When a window is non-host triggered, starting address of a window buffer is calculated as below.
  - non-YUV-planar mode

    starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET
  - YUV-planar mode:

    Y-starting-address = START_ADDR + BUF_STRIDE * buf_index + ADDR_V_OFFSET * LINE_STRIDE + ADDR_H_OFFSET

    U-starting-address = START_ADDR_U + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    V-starting-address = START_ADDR_V + UV_BUF_STRIDE * buf_index + ADDR_V_OFFSET * UV_LINE_STRIDE / denom1 + ADDR_H_OFFSET / denom2

    where denom1/denom2 are equal to 1 or 2 depending on the actual planar format, derived natively by hardware.

    buf_index is the index transmitted by the triggering module pointing to the first buffer of a frame.

Programming Restrictions

- For tiled address mode:

  Image surface can only aligned to multiples of 256, thus the following restrictions:

- START_ADDR, START_ADDR_U, START_ADDR_V need to be multiples of 256.
- BUF_STRIDE, UV_BUF_STRIDE need to be multiples of 256
- LINE_STRIDE, UV_LINE_STRIDE need to be multiples of 16
- ADDR_H_OFFSET needs to be even in yuv planar format, or a multiple of bytes per pixel in other formats. If H_DIRECTION=DECREMENT, however, it should point to last valid byte, which is an odd offset..
- ADDR_V_OFFSET needs to be multiple of 2 in YUV planar format, unless H_DIRECTION=DECREMENT, but with no restrictions on other color formats

- For linear address mode:

  Image surface can be aligned 2, 4, or 8 bytes, depending on the color formats.

  As an additional restriction for display, START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16. When a surface is not aligned to 16 bytes, program START_ADDR with the memory surface address with its least 4 significant bits zeroed out and add these 4 address bits to the original H_OFFSET. (So the formulae in Starting Address Calculation above still hold)

  - For all formats:

    START_ADDR, START_ADDR_U and START_ADDR_V need to be multiples of 16.

    For 16-bpp formats,

    (START_ADDR+H_OFFSET) need to be a multiple of 2.

    For 32-bpp formats,

    (START_ADDR+H_OFFSTE) needs to be a multiple of 4.

  - For YUV planar formats:

    BUF_STRIDE, UV_BUF_STRIDE:

    BUF_STRIDE[2:1]=UV_BUF_STRIDE[1:0]

    or as a stricter constraint: BUF_STRIDE be a multiple of 8, UV_BUF_STRIDE be a multiple of 4.

    LINE_STRIDE, UV_LINE_STRIDE:

    LINE_STRIDE and UV_LINE_STRIDE need to be at least 16.

    LINE_STRIDE needs to be a multiple of 8, UV_LINE_STRIDE needs to be a multiple of 4.

    ADDR_H_OFFSET: Needs to be even unless H_DIRECTION=DECREMENT, in which case should point to the last byte pixel, which is at an odd offset.

    ADDR_V_OFFSET: Needs to be even unless V_DIRECTION=DECREMENT, in which case should point to the last valid line, which is at an odd offset

- Memory allocation for non-host triggered case:

  When a window buffer is not controlled by host (software) then a frame may be stored in multiple buffers. In this case, the buffers must be contiguous in the memory because display will use the same luma or chroma line strides for all lines in the frame.

  Also buffer wraparound must not occur in the middle of the displayed part of the frame. The controlling module will send frame start and frame end indicators (flags) to display module to indicate the beginning and end of frame. Buffer start address is latched when frame start flag is active but the actual buffer start address is not switched until frame end flag is active. In the case where one buffer corresponds to one frame then frame start and frame end flag are active every time a buffer index is sent.

Offset: 0x800 │ Byte Offset: 0x2000 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR: Window C Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-

Bit	Reset	Description
		planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

25.14.2 DC_WINBUF_C_START_ADDR_NS_0

Window C Shadowed Start Address

Offset: 0x801 | Byte Offset: 0x2004 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_NS: Window C Shadowed Start Address. This is ARM set shadow of Start Address.

25.14.3 DC_WINBUF_C_START_ADDR_U_0

Window C Start Address for U Plane

Offset: 0x802 | Byte Offset: 0x2008 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_U: Window C Start Address for U plane. This is a byte address.

25.14.4 DC_WINBUF_C_START_ADDR_U_NS_0

Window C Shadowed Start Address for U Plane

Offset: 0x803 | Byte Offset: 0x200c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_U_NS: Window C Shadowed Start Address for U plane. This is ARM set shadow register of U start address

25.14.5 DC_WINBUF_C_START_ADDR_V_0

Window C Start Address for V Plane

Offset: 0x804 | Byte Offset: 0x2010 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_V: Window C Start Address for V plane. This is a byte address.

25.14.6 DC_WINBUF_C_START_ADDR_V_NS_0

Window C Shadowed Start Address for V Plane

Offset: 0x805 | Byte Offset: 0x2014 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_START_ADDR_V_NS: Window C Shadowed Start Address for V plane. This is ARM set shadow register of U start address

25.14.7 DC_WINBUF_C_ADDR_H_OFFSET_0

Window C Horizontal Address Offset

Offset: 0x806 | Byte Offset: 0x2018 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_ADDR_H_OFFSET: Window C Horizontal address offset. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies horizontal offset of Y plane. The horizontal offsets of U/V plane is derived by hardware.

25.14.8 DC_WINBUF_C_ADDR_H_OFFSET_NS_0

Window C Shadowed Horizontal Address Offset

Offset: 0x807 | Byte Offset: 0x201c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_ADDR_H_OFFSET_NS: Window C Shadowed Horizontal address offset. This is ARM set shadow of ADDR_H_OFFSET

25.14.9 DC_WINBUF_C_ADDR_V_OFFSET_0

Window C Vertical Address Offset

Offset: 0x808 | Byte Offset: 0x2020 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_ADDR_V_OFFSET: Window C Vertical address offset. This is the vertical coordinate. For YUV planar pixel format, this specifies the vertical coordinate of Y plane. The vertical coordinate of U/V plane is derived by hardware.

25.14.10 DC_WINBUF_C_ADDR_V_OFFSET_NS_0

Window C Shadowed Vertical Address Offset

Offset: 0x809 | Byte Offset: 0x2024 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_ADDR_V_OFFSET_NS: Window C Shadowed Vertical address offset. This is ARM set shadow of ADDR_V_OFFSET

25.14.11 DC_WINBUF_C_UFLOW_STATUS_0

Window C FIFO Underflow Status Register

Offset: 0x80a | Byte Offset: 0x2028 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	COUNT_OFLOW: Flag bit that indicates that the underflow event counter has overflowed. There were too many events. If COUNT_OFLOW is set, UFLOW_COUNT is meaningless. Cleared on write.

Bit	Reset	Description
23:0	X	UFLOW_COUNT: Underflow count. This field indicates the number of contiguous groups of output pixels for which there was no data in the FIFO. For example, if the valid from the FIFO is low for 10 consecutive cycles and then goes high, the counter will increment by one. Reset to zero on write.

25.14.12 DC_WINBUF_C_SURFACE_WEIGHT_0

Window C Surface Weights

Offset: 0x80c │ Byte Offset: 0x2030 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
6:5	X	C_SURFACE_WEIGHT_V: Window C V Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
4:3	X	C_SURFACE_WEIGHT_U: Window C U or UV Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
2:1	X	C_SURFACE_WEIGHT_Y: Window C Y or packed Surface Weights. 00 - Weight is 2 01 - Weight is 4 10 - Weight is 8 11 - Weight is 16 0 = WEIGHT_2 1 = WEIGHT_4 2 = WEIGHT_8 3 = WEIGHT_16
0	0x0	C_SURFACE_WEIGHT_OVERRIDE: Weight override 0 = DISABLE 1 = ENABLE

25.14.13 DC_WINBUF_C_START_ADDR_HI_0

Window C Higher 32 bits of Start Address

Makes the start address 64 bits by adding the higher 32 bits below.

Offset: 0x80d │ Byte Offset: 0x2034 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	C_START_ADDR_HI: Window C Start Address. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSBs are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

25.14.14 DC_WINBUF_C_START_ADDR_HI_NS_0

Window C Shadowed Higher 32 bits of Start Address

Offset: 0x80e │ Byte Offset: 0x2038 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	C_START_ADDR_HI_NS: Window C Shadowed Start Address. This is ARM set shadow of Start Address.

25.14.15 DC_WINBUF_C_START_ADDR_HI_U_0

Window C Higher 32 bits of Start Address for U Plane

Offset: 0x80f │ Byte Offset: 0x203c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	C_START_ADDR_HI_U: Window C Start Address for U plane. This is a byte address.

25.14.16 DC_WINBUF_C_START_ADDR_HI_U_NS_0

Window C Shadowed Higher 32 bits of Start Address for U Plane

Offset: 0x810 │ Byte Offset: 0x2040 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	C_START_ADDR_HI_U_NS: Window C Shadowed Higher 32 bits of Start Address for U plane. This is ARM set shadow register of U start address

25.14.17 DC_WINBUF_C_START_ADDR_HI_V_0

Window C Higher 32 bits of Start Address for V Plane

Offset: 0x811 │ Byte Offset: 0x2044 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	C_START_ADDR_HI_V: Window C Higher 32 bits of Start Address for V plane. This is a byte address.

25.14.18 DC_WINBUF_C_START_ADDR_HI_V_NS_0

Window C Shadowed Higher 32 bits of Start Address for V Plane

Offset: 0x812 │ Byte Offset: 0x2048 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	C_START_ADDR_HI_V_NS: Window C Shadowed Higher 32 bits of Start Address for V plane. This is ARM set shadow register of U start address

25.14.19 DC_WINBUF_C_UFLOW_CTRL_0

When the underflow debug mode is enabled. Instead of sending out the last valid pixel, DBG_PIXEL is sent out.

Offset: 0x824 | Byte Offset: 0x2090 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	C_UFLOW_CTRL_DBG_MODE: 0 = DISABLE 1 = ENABLE

25.14.20 DC_WINBUF_C_UFLOW_DBG_PIXEL_0

Offset: 0x825 | Byte Offset: 0x2094 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	C_UFLOW_DBG_PIXEL

25.14.21 DC_WINBUF_C_UFLOW_THRESHOLD_0

Offset: 0x826 | Byte Offset: 0x2098 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	C_UFLOW_THRESHOLD

25.14.22 DC_WINBUF_C_SPOOL_UP_0

Spool up time configuration for different windows related logic.

SPOOL_UP_CTRL = MAX - This is the current Tegra 4 behavior. This allows for memfetch to start fetching for a window (irrespective of its relative position on the screen), at the very beginning of the VBlank of the next frame. Since the spool up time is maximum, this should have the least probability of underflow of the display pipe. However, on the down side, since all three windows would start fetching at the same time, and since display is a high priority client for the MC, it has a high potential to starve out other clients.

SPOOL_UP_EDGE :- By default the memfetch start signal generated by the hvtgen logic is one line clock wide.

For SPOOL_UP_CTRL = MAX:

- If SPOOL_UP_EDGE is programmed to NEGEDGE, it ensures that if any act_req is sent during the time vcounter = 0 (at the beginning of the frame). The activate value is loaded into the active state and the memfetch would see the latest updated value.

  The downside of programming SPOOL_UP_EDGE to NEGDGE is 1 line clock worth of spool up is lost. This is required based upon Tegra compatibility.
- If SPOOL_UP_EDGE is programmed to POSEDGE, the memfetch starts fetching at the very beginning of the frame start pulse (1 line clock wide) and gives maximum spool up time. However, this may give test failure and potential Tegra compatibility issues as described above.

Offset: 0x827 | Byte Offset: 0x209c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
28:16	X	C_SPOOL_UP_DURATION
1	0x0	C_SPOOL_UP_EDGE: 0 = NEGEDGE 1 = POSEDGE

Bit	Reset	Description
0	0x0	C_SPOOL_UP_CTRL. This bit should be set to MAX for Tegra 4 devices. 0 = MAX 1 = PROGRAMMABLE

25.14.23 DC_WINBUF_C_SCALEFACTOR_THRESHOLD_0

Registers for latency allowance scaling. Threshold is in terms of 64 byte atoms. Display memfetch compares its buffer occupancy with the threshold and generates indication to the MC if its buffer occupancy is above HWM or below LWM.

Offset: 0x828 │ Byte Offset: 0x20a0 │ Read/Write: R/W │ Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:16	0xffff	C_SF_LWM_THRESHOLD
15:0	0xffff	C_SF_HWM_THRESHOLD

25.14.24 DC_WINBUF_C_LATENCY_THRESHOLD_0

Register for ready for latency event like DVFS. Threshold is in terms of 64 byte atoms. Display memfetch compares its buffer occupancy with the threshold and generates indication to the MC if its buffer occupancy is above the threshold.

Offset: 0x829 │ Byte Offset: 0x20a4 │ Read/Write: R/W │ Reset: 0x0000ffff (0bxxxxxxxxxxxxxxxx1111111111111111)

Bit	Reset	Description
15:0	0xffff	C_RDY4LATENCY_THRESHOLD

25.14.25 DC_WINBUF_C_MEMFETCH_DEBUG_STATUS_0

Memfetch Debug Status

Offset: 0x82a │ Byte Offset: 0x20a8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15	0x0	B_ALIGN_FIFO_NON_IDLE: align FIFO idle at end of frame. 0= align FIFO is idle at end of frame. 1= align FIFO is not idle at end of frame
14	0x0	B_PIPE1_FIFO_NON_IDLE: pipe1 FIFO idle at end of frame. 0= pipe1 FIFO is idle at end of frame. 1= pipe1 FIFO is not idle at end of frame
13	0x0	B_PIPE0_FIFO_NON_IDLE: pipe0 FIFO idle at end of frame. 0= pipe0 FIFO is idle at end of frame. 1= pipe0 FIFO is not idle at end of frame
12	0x0	B_FRAME_FIFO_NON_IDLE: frame FIFO idle at end of frame. 0= frame FIFO is idle at end of frame. 1= frame FIFO is not idle at end of frame
11	0x0	B_SURFACE_FIFO_NON_IDLE: surface FIFO idle at end of frame. 0= surface FIFO is idle at end of frame. 1= surface FIFO is not idle at end of frame
10	0x0	B_TAG_FIFO_NON_IDLE: tag FIFO idle at end of frame. 0= tag FIFO is idle at end of frame. 1= tag FIFO is not idle at end of frame
9	0x0	B_RSP_FIFO_NON_IDLE: Response FIFO idle at end of frame. 0= response FIFO is idle at end of frame. 1= response FIFO is not idle at end of frame
8	0x0	B_REQ_FIFO_NON_IDLE: Request FIFO idle at end of frame. 0= request FIFO is idle at end of frame. 1= request FIFO is not idle at end of frame
7	0x0	B_DP_VALID_NON_IDLE: DP lines valid at end of frame. 0= DP lines are not valid at end of frame. 1= DP lines are valid at end of frame
6	0x0	B_V_SSM_NON_IDLE: V SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame

Bit	Reset	Description
5	0x0	B_U_SSM_NON_IDLE: U SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
4	0x0	B_Y_SSM_NON_IDLE: Y SSM is non-idle at end of frame. 0= SSM is idle at end of frame. 1= SSM is not idle at end of frame
3	0x0	B_DP_POOL_AVAIL: DP thinks there is data in the pool at end of frame. 0= No pool data available at end of frame. 1= pool data available at end of frame
2	0x0	B_POOL_NOT_EMPTY: Status of pool at end of frame. 0= pool is empty at end of frame. 1= pool is not empty at end of frame
1	0x0	B_UNDERFLOW_LINE1: Underflow of line0. 0= No underflow. 1= line0 underflowed
0	0x0	B_UNDERFLOW_LINE0: Underflow of line0. 0= No underflow. 1= line0 underflowed

25.14.26 DC_WINBUF_C_MEMFETCH_CONTROL_0

Memfetch Control Register

Offset: 0x82b │ Byte Offset: 0x20ac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	C_MEMFETCH_CLK_GATE_OVERRIDE: Disable clock-gating of memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE
0	0x0	C_MEMFETCH_RESET: Reset memfetch for a controller (all three windows) 0 = DISABLE 1 = ENABLE

25.14.27 DC_WINBUF_C_OCCUPANCY_THROTTLE_0

Offset: 0x82c │ Byte Offset: 0x20b0 │ Read/Write: R/W │ Reset: 0xffff0000 (0b1111111111111111xxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:16	0xffff	C_OCCUPANCY_MAX_THRESHOLD
0	0x0	C_OCCUPANCY_THROTTLE_MODE: 0 = DISABLE 1 = ENABLE

25.14.28 DC_WINBUF_C_SCRATCH_REGISTER_0_0

Offset: 0x82d │ Byte Offset: 0x20b4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	C_SCRATCH_REGISTER_0: Scratch register 0

25.14.29 DC_WINBUF_C_SCRATCH_REGISTER_1_0

Offset: 0x82e │ Byte Offset: 0x20b8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

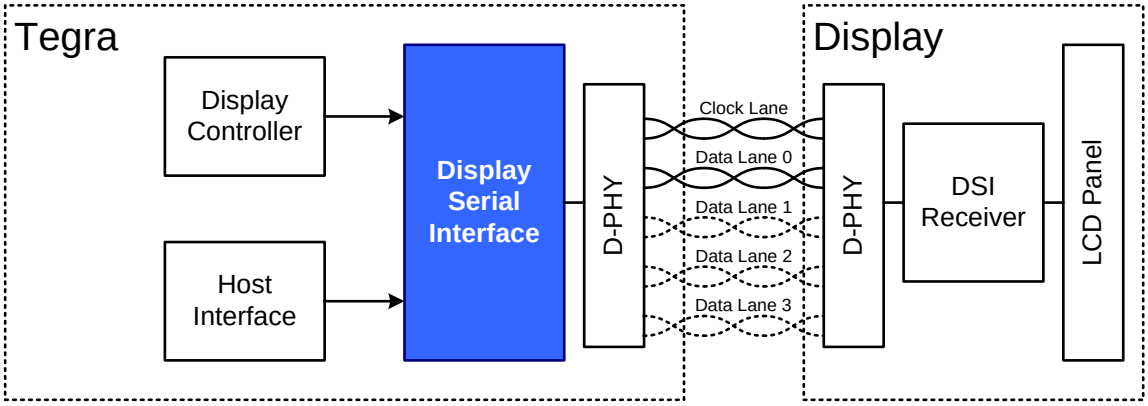
Bit	Reset	Description
31:0	0x0	C_SCRATCH_REGISTER_1: Scratch register 1

26.0 MIPI-DSI (DISPLAY SERIAL INTERFACE)

PRELIMINARY: Pending Review

The Display Serial Interface (DSI) is a MIPI standard serial bitstream that provides a low pin-count interface to a display panel. DSI reduces both package pin-count and I/O power consumption compared with parallel solutions. It transfers pixel data from either one of the display controllers (internal to the Tegra® 4 device) to an external third-party LCD module. The physical positioning of the DSI module in relation to other units / devices in the system is shown in Figure 106.

Figure 106: System Block Diagram



DSI is a replacement for the MIPI DPI and DBI standards and follows the use cases of these interfaces. From a data transport point of view, MIPI DPI is an interface similar to a traditional raster-based isochronous display interface, and DBI is an asynchronous packet based transfer mechanism. DSI behaves like MIPI DPI when in Video Mode, and implements a Command Mode to handle the DBI interface. Any implementation must implement both these modes of operation.

26.1 Functionality

26.1.1 Display Controller Interface

The interface between the display controller and the DSI unit consists of pixel data, Sync data, and a Line-Type code.

DSI captures the state of the line type code every line. This information is then used to look up a pre-programmed packet sequence that corresponds to that line type. There are several different line types to select from.

26.1.2 Packet Poster

The packet poster determines which packets will be sent on what video line when the DSI is in any of the 3 video modes and is being controlled by the display controller. The functions that this sub-unit performs are:

- Line type look-up
- Packet sequence generation
- Pixel data / packet stream integration

There are currently 6 valid line types that can be encoded on the 3 bits from the display controller. The other 2 line types are reserved for future use. The line type coding is shown in the following table.

Table 74: Line Type Descriptions

Line Type	Description
0	Blank line starting with VS
1	Blank line starting with VE
2	Blank line starting with HS
3	Active line
4	First blank line after active
5	First active line
6, 7	Reserved

The packet sequences corresponding to each line type are held in a pair of 32-bit registers. Each register holds three packet descriptors, and each packet descriptor has the following information:

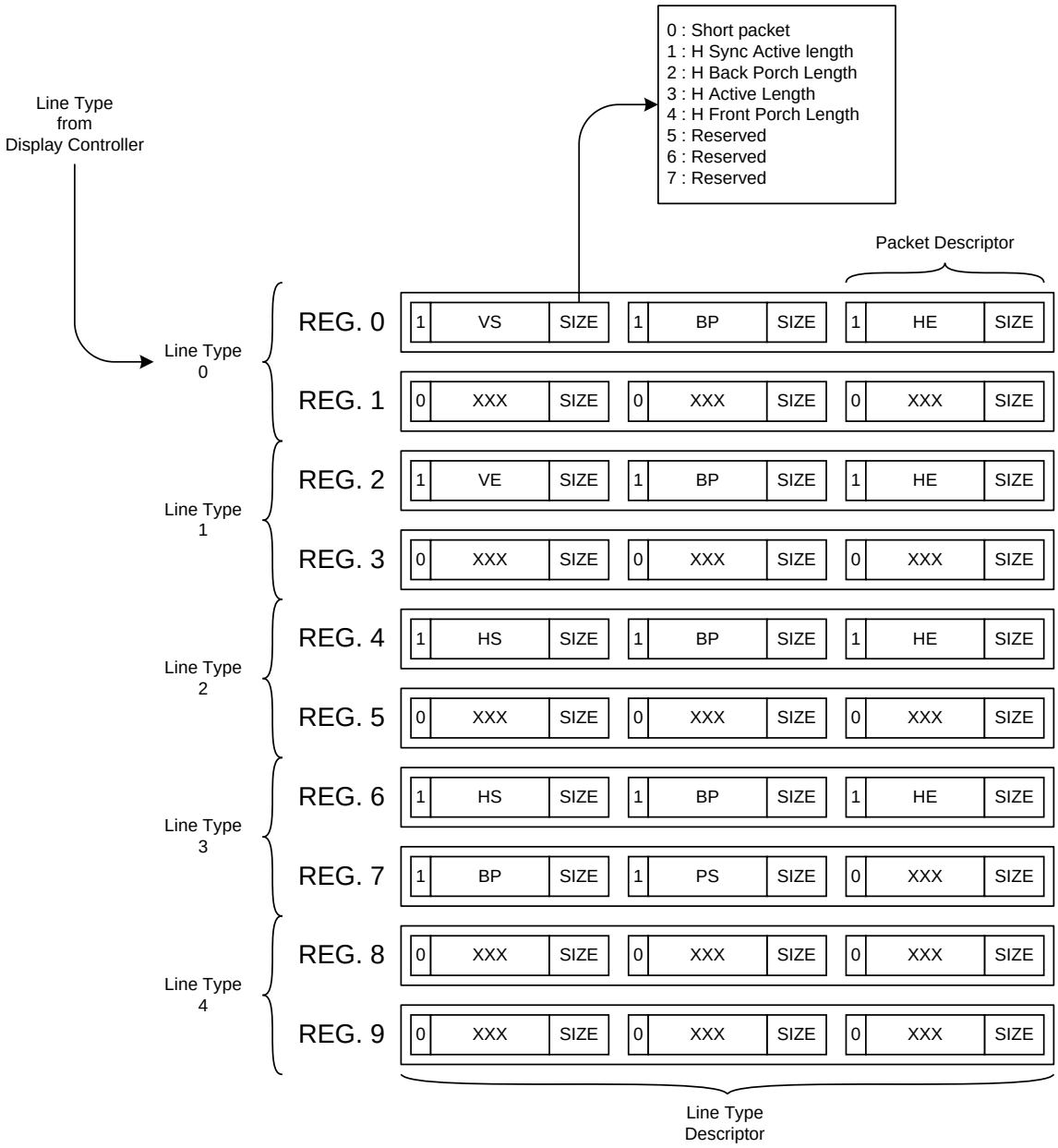
- Valid bit
- 6-bit packet ID
- 3-bit length index

The valid bit simply states whether or not a packet should be generated from that descriptor. A zero valid bit implies that no more packets in this packet sequence should be generated. Thus, the valid bit acts like a null terminator for the list of packet descriptors.

The packet ID is simply the MIPI DSI packet ID value. Software can therefore program any sequence of packets required for that line type.

Rather than have a 16-bit packet length field for every packet descriptor, the length is replaced by a 3-bit index which addresses the full 16-bit length for that packet type. This length is held elsewhere in separate length registers. A complete picture of the packet description data structures is shown in the following figure. This diagram shows typical packet mnemonics for the packets to be generated for video mode with sync pulses. By changing the programming of the packet sequences, any of the video modes, including burst mode, can be realized.

Figure 107: Line Type Look-up and Packet Sequence Descriptors



26.1.3 Host Interface

While many displays use a raster-based packet stream, in some applications it is more efficient to have a frame store in the display device itself and to only update the contents of the frame store when the required pixels have changed. For these types of displays, there is a software-accessible interface for sending command packets using the DSI command mode operation. This mode should also be used to access control and status registers within the display device.

The Host interface consists of a small FIFO for holding packet information and a special register for writing data into the FIFO. In this way, several packets may be queued up by software and sent in one HS burst. This is more efficient that sending the packets one at a time.

When very long sequences of packets are required to be sent – for example, when the host interface is emulating video mode, or where a large amount of data needs to be sent to the display device in one packet – it is possible for the host to use the large video line-store FIFO.

26.1.4 Clocking

The clocks in the DSI include:

- Host Clock = 275 MHz {Maximum frequency}
- Pixel Clock = 2.49 to 300 MHz for all modes.
- Application/Lane Management Byte Clock = function of lanes, pixel clock, pixel depth. 10 to 125 Mbps
- Fixed LP receive clock = 20 to 216 MHz
- Lane Bit clock = 40 to 500 MHz differential clock, DDR data (80 to 1000 bps).

More details on pixel clock/byte_clock selection are explained in the next section.

Pixel Clock / DSI Byte Clock Ratios

When running the DSI interface in one of the two non-burst video modes, the relationship between the DSI D-PHY bit-rate clock, the DSI byte-rate lane clock, and the display controller pixel clock must be exact. This will ensure that precise raster timing information is conveyed to the display peripheral.

There is always a fixed ratio of 8:1 between the D-PHY bit clock and the lane management layer byte clock since there are 8 bits in each byte that are serialized. Therefore, only relationships between pixel clock and byte clock and between pixel clock and bit clock will be discussed.

The relationship between pixel clock and byte clock is shown in Table 75. As can be seen, for some modes, these ratios are not simple powers of 2.

Table 75: Pixel Clock: Byte Clock for Various Modes

Lanes	Pixel Data Format			
	16 bpp	18 bpp (packed)	18 bpp	24 bpp
1	1:2	4:9	1:3	1:3
2	1:1	8:9	2:3	2:3
3	3:2	12:9	1:1	1:1
4	2:1	16:9	4:3	4:3
5	5:2	20:9	5:3	5:3
6	3:1	8:3	2:1	2:1
7	7:2	28:9	7:3	7:3
8	4:1	32:9	8:3	8:3

Because the pixel clock is effectively derived from the D-PHY bit clock, it can be more instructive to look at the number by which you must divide the bit clock to get the correct pixel clock. This information is shown in Table 76. Here, the numbers look reasonable except for the problematic cases of 16bpp over a 3-lane link and 18 bpp (packed) over a 4 lane link. For these two situations, you cannot simply divide the bit rate clock by some integer to arrive at the pixel clock. Moreover, if you need a 50:50 duty cycle for the pixel clock, then you will need to toggle the pixel clock every N/2 bit clock cycles, where N is the number in Table 76. However, if you do this, 18 bpp (packed) over 2 lanes also becomes a problem.

Table 76: Value to Divide Bit Clock by to Get Pixel Clock

Lanes	Pixel Data Format			
	16 bpp	18 bpp (packed)	18 bpp	24 bpp
1	16	18	24	24
2	8	9	12	12

Lanes	Pixel Data Format			
	16 bpp	18 bpp (packed)	18 bpp	24 bpp
3	16 / 3	6	8	8
4	4	9 / 2	6	6
5	16/5	18/5	24/5	24/5
6	8/3	3	4	4
7	16/7	18/7	24/7	24/7
8	2	9/4	3	3

Table 77: Value of Shift Clock Divider to Get Pixel Clock (Non-Burst Mode)

Lanes	Data Format (16 bpp)	Data Format (18 bpp)	Data Format (24 bpp)
1	14	16	22
2	6	7	10
3	3.333334	4	6
4	2	2.5	4
5	1.6	1.8	2.4
6	0.666667	1	2
7	1.14285714	1.285714286	1.714285714
8	0	0.25	1

Note: The fractional values in the above table cannot be supported through configuration fields because of a possible limitation in programming the shift divider for deriving the pixel clock from the byte clock in non-burst mode, that is, the SHIFT_CLK_DIVIDER field of the DC_DISP_DISP_CLOCK_CONTROL_0 field. This limitation occurs only in non-burst video mode.

To resolve these issues, derive the display controller pixel clock from a separate PLL and then lock that PLL to the DSI D-PHY PLL using a phase comparator and the appropriate divide ratios obtained from the tables.

DSI PLL – Clock Mux

In the Tegra 4 device, the DSI implements a clock mux for the clock selection of the write clock domain to the line buffer and its upstream path in the DSI.

A single DSI instance receives a pixel data stream from either of the display controller heads. The single bit register field dsi_vid_src in the dsi_clk domain selects the active display head. The selected display pixel clock provides the necessary clocking to the front end of the DSI.

In cases where the Host transmission uses the line buffer, dsi_clk can provide the necessary clocking using the single bit register field pkt_wr_fifo_sel.

26.2 Modes of Operation

The DSI operates in two transmission modes: Video and Host/Command

26.2.1 Video Mode

Communication with the peripheral is by isochronous data transfer similar to a typical video interface. There is no Bus Turn Around permitted in Video Mode, though there is a mandatory period of LP operation at least once per frame. There are three different sub-modes of Video Mode. These are outlined below:

Non-Burst Mode with Start and End

In this mode, the DSI must match the timing of a traditional video raster as closely as possible. This means all information about the start and end of parameters like vertical and horizontal sync, front and back porches must be conveyed to the receiving peripherals via the timing of the transmission of the High-Speed sync packets.

Non-Burst Mode without Start and End

This is like Non-Burst Mode with Start and End, except that the Sync-Active and Sync-End packets are not sent. Pixels must still be delivered at the correct rate.

Burst Mode

Only the timing of Sync-Start packets is required to be accurately conveyed in this mode. The data rate of the pixel data is arbitrary and is typically higher than the pixel rate of the peripheral. This allows time to transmit other information during the periods where no more pixels are needed.

26.2.2 Host/Command Mode

Communication with the peripheral is by asynchronously timed and variable length packets. The packets may contain video data (Display Controller/Host) or control data. In Command Mode, return (read) data can be requested from the peripheral. The DSI will issue a Bus Turn Around (BTA) request and relinquish control of the bus to the peripheral.

26.3 FIFO Buffers

26.3.1 Video Mode Operation

26.3.1.1 Writing Data

The following packets are written into the large data FIFO based on the various triggering events described.

Table 78: FIFO Trigger Events

Trigger Event	Packet
Leading edge of Vertical Sync.	VSync Start
Trailing edge of Vertical Sync	VSync End
Leading edge of Horizontal Sync, *unless* simultaneous with leading or trailing edge of vertical sync	HSync Start
Trailing edge of Horizontal Sync	HSync End
Immediately following VS, HS, VE or HE packets. Packet Word Count determined by display controller timing parameters.	Blanking packet
Immediately following blanking packet	Pixel Stream packet (16, 18, or 24 bits/pixel)

26.3.1.2 Reading Data

The initiation of the reading of the FIFO is triggered by a delayed version of the horizontal sync pulse from the display controller. The delay should be sufficient that the FIFO will not completely drain as reading of the FIFO continues. However, the delay should not be so long as to cause the FIFO to overflow with data.

26.3.2 Command Mode Operation

26.3.2.1 Writing Data from the Display Controller

Upon receipt of the leading edge of Vertical Sync, the display controller sends whatever DCS command packets are required to initialize the display to receive pixels. Then, as pixels start to arrive from the display controller, a DCS Long Write packet is sent – one per line – which contains the pixel data for that line. No synchronization or blanking packets should be sent. Data packets continue to be sent until the end of the active period. Like video mode packets, there is no need to calculate ECC or CRC words, as this is performed by the hardware on the other side of the data FIFO.

The Packet Sequence registers behave in the same manner as they do for Video Mode packet generation with the following exceptions:

- For Long packets, a DCS command ID byte must be inserted as the first byte of the data payload of the packet, ahead of the pixel data. The packet header Word Count must be 1 more than the number of bytes in the pixel data to take into account this extra byte. It is the responsibility of software to make sure the packet length is programmed correctly.
- For Line Type 4, any data contained in the Host Data FIFO should be appended to the end of any packets defined in the Packet Sequence for Line Type 4. In this way, the Host may send DCS commands to the Display Peripheral while the Display Controller is sending pixel information to the Display Peripheral.

26.3.2.2 Writing Data from the Host

The host should choose which data FIFO – small or large – to use prior to sending packet information. The large data FIFO is only available if the display controller is not currently using it. The software is responsible for constructing whatever packets are required for the desired operation and should be written into the FIFO via the host. There is no need for the software to compute ECC or CRC words, because this is performed by the hardware on the other side of the data FIFO.

26.3.2.3 Reading Data

The initiation of the reading of data from the FIFO is triggered by a FIFO threshold. Once the threshold is reached, the D-PHY starts to drain the FIFO, and the packets are sent to the display. The threshold should be set such that there is no danger of either FIFO overflow from excessive data from the host or display controller, or FIFO underflow caused by a lack of sufficient data in the FIFO.

26.3.3 FIFO Sizing

This subsection provides the threshold calculations for video non-burst and non-burst with sync ends modes.

- InRate = pixClk (pixel clock)
- OutRate = (PhyClk x LaneNum) / BytePerPix
- Length = Input frame width (length in pixels)
- Threshold = Minimum FIFO depth to start transfer of data to CIL
- Length/OutRate > (Length – Threshold) / InRate
- Threshold > Length * (1 – InRate / OutRate)

Table 79: Output Rates and Thresholds for Width = 800, PhyClk = 128 MHz

Data Format		Number of Lanes			
Bytes per Pixel		**1**	**2**	**4**	**8**
16	Out Rate	62.5	250	10000	4000
	Threshold	199.68	343.2	385.8	396.45
18 (Packed)	Out Rate	55.55	222.22	888.88	3555.55
	Threshold	207.72	378.1	432	445.5
24/18 (Loose)	Out Rate	41.66	166.66	666.66	2666.66
	Threshold	209.28	472.2	568	592

26.4 Programming Guidelines

The packet timing diagrams in this section use the following key:

Figure 108: Video Mode Timing Diagram Key

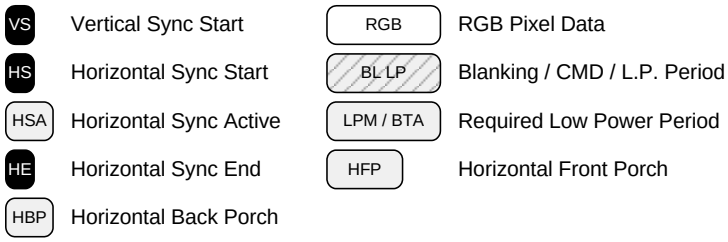


Table 80: Timing Diagram Parameter List

Parameter	Description
tHBP	Horizontal Back Porch period
tHACT	Horizontal Active period
tVBP	Vertical Back Porch period
tVACT	Vertical Active period
tVFP	Vertical Front Porch period
tL	Total line period

26.4.1 Initialization

The start-up and shut-down procedures vary depending on what mode of operation is required. Examples of programming sequences for the main modes are given below.

26.4.1.1 PLLD Programming

The DSI PLL is called PLLD in the Tegra 4 devices. Just like the other PLLs in the system, the PLLD has M, N, and P registers for programming the various divide ratios for the clock. When used to drive the Display pixel clock and DSI byte clock, the P divider is not used. Instead, there is a second divider that is used specifically for obtaining the Display pixel clock. The division ratio is labeled as D in the code below, but the correct name of the register should be used when programming the PLL.

```
//**** Environment ... ****

const float Fomin = 80.0;   // Min. oscillator frequency (MHz)
const float Fomax = 1000.0; // Max. oscillator frequency (MHz)
const float Fcmin = 1.0;    // Min. PLL phase comparison frequency (MHz)
```

```
const float Fcmax = 6.5;    // Max. PLL phase comparison frequency (MHz)

// MIPI Pixel formats ...
typedef enum { BIT16P, BIT18P, BIT18NP, BIT24P } pixel fmt;

//**** PLL Programming Information ... ****

int   M;         // Value by which the Reference clock is divided
                 // to get the comparison frequency
int   N;         // Value by which the Oscillator clock is divided
                 // to get the comparison frequency
int   D;         // Ratio between D-PHY bit clock and pixel clock

float Fosc;      // PLL oscillator frequency, in MHz   (information only)
float Fcomp;     // Phase comparator frequency, in MHz (information only)


//**** Function that computes the values of M, N and D ... ****

bool calc M N D(float Fref, float Fpixel, float tolerance,
                pixel fmt fmt, int lanes)

/******************
  Fref      : Input Reference frequency - in MHz
  Fpixel    : Desired output pixel clock frequency
  tolerance : Fractional error allowed in the output frequency
  fmt       : Pixel format. One of BIT16P, BIT18P, BIT18NP and BIT24P
  lanes     : Number of DSI lanes in use
  Function returns "true" on success and "false" otherwise
*******************/

{
  float Nfrac;
  bool  done = false;

  // Compute the value of D ...

  switch (fmt) {
    case BIT16P  : D = 16; break;
    case BIT18P  : D = 18; break;
    case BIT18NP :
    case BIT24P  : D = 24; break;
    }

  D = D / lanes;
  Fosc = Fpixel * clk ratio; // Calculate the Oscillator Frequency

  // Compute the values of M and N ...

  if ((Fosc > Fomin) && (Fosc < Fomax)) {
   // If the oscillator frequency is "legal" ...
    M = (int)(0.99 + Fref / Fcmax);
    do {
      // For each value of M, compute the comparison frequency
```

```
      Fcomp = Fref / (float)(M);
      // Now compute the exact value of N ...
      Nfrac = Fosc / Fcomp;
      // Strip off the integer and fractional parts ...
      N     = (int)(Nfrac);
      Nfrac = Nfrac - (float)(N);
      if (N > 0) {
        if ((Nfrac / N) < tolerance)
          // If the error is less than the tolerance, then we're done
          done = true;
        else
          // Too much error, so try dividing the reference some more ...
          M++;
        }
      else // N too small, so divide the reference by some more ...
        M++;
      } while (!done && (M < 32) && (Fcomp > Fcmin));
    }

  return (done);
}
```

26.4.1.2 Video Mode Start-up

All 3 video modes – Burst, Non-Burst and Non-Burst with Sync Ends –have the same start-up sequence with a slight variation between Burst and Non-Burst modes:

- Set up the clocks. This involves configuring and enabling the DSI PLL (PLLD). For Non-Burst and Non-Burst with Sync End modes, the Display Controller must also be programmed to take its pixel clock from the DSI PLL. For Burst Mode, the Display Controller can take its pixel clock either from the DSI PLL or from another clock source. Program the PLLD registers with the correct OSC_FREQ and program the PLLD_BASE register. In addition, the PLLP must be running in order to provide the LP receive clock. Program the PLLP registers with the correct OSC_FREQ and program the PLLP_BASE register.
- Depending on the sub-mode, the various Packet Sequence registers and Packet Length registers must be programmed.
- Set the `VID_TX_TRIG_SRC` field in the `DSI_CONTROL` register to `SOL` to select which display controller will source the video data.
- Enable the DSI
- Program the display controller to produce the raster size required
- Enable the display controller

When the display controller starts sending SOL and data, the DSI automatically starts creating and transmitting packets to the display device.

26.4.1.3 Command Mode Start-up

- Set up the clocks. Program and enable the DSI PLL (PLLD). In addition, the PLLP must be running in order to provide the LP receive clock. Program the PLLP registers with the correct OSC_FREQ and program the PLLP_BASE register. Unlike the Video modes, the selection of the PLLD output clock frequency is essentially arbitrary. The clock frequency should be based on the expected data throughput and the requirements of the particular display device. Program the PLLD registers with the correct OSC_FREQ and program the PLLD_BASE register. In addition, the

PLLP must be running in order to provide the LP receive clock. Program the PLLP registers with the correct OSC_FREQ and program the PLLP_BASE register.

- Set the HOST_TX_TRIG_SRC field in the HOST_DSI_CONTROL registers to `IMMEDIATE`. Set any other state required.
- Enable DSI.

The DSI should now be ready to accept command packets written to the DSI_WR_DATA register.

26.4.2 Video Mode Programming

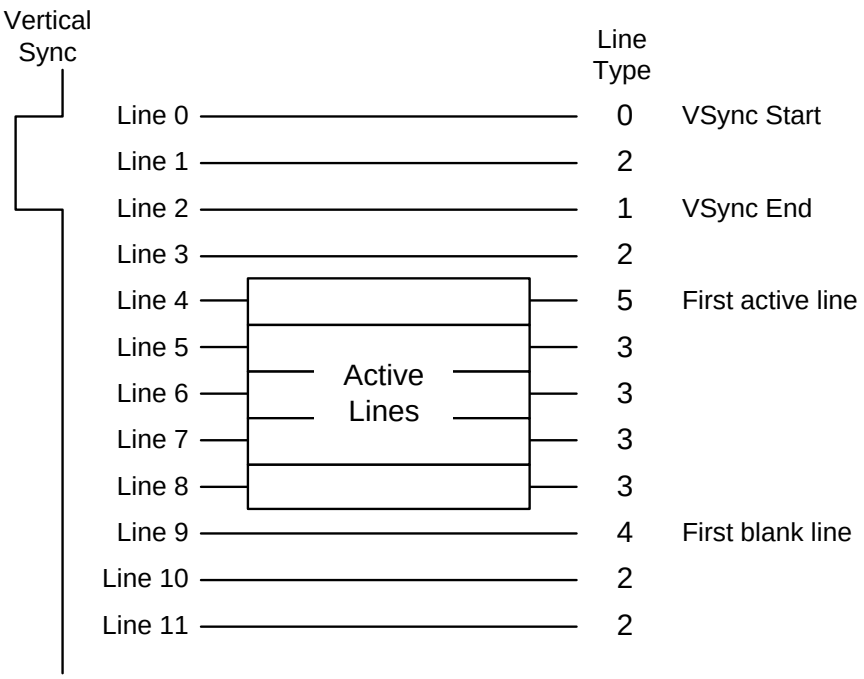
The programming model has been designed to accommodate current video mode sequences and potentially future variation. Each line of video output is associated with a “line type”. This line type is automatically generated by the display controller hardware according to the following table.

Table 81: Line Type Descriptions

Line Type	Description	Typical raster position
0	Blank line starting with VS	First line of the raster
1	Blank line starting with VE	Line corresponding to Vsync end.
2	Blank line starting with HS	Vertical blanking line
3	Active line starting with HS	Active line
4	First blank line after active	First blank line after active
5	First active line	First active line
6, 7	Reserved	

Figure 109 illustrates a typical raster showing the relationship between various events in the raster and the line types.

Figure 109: Example Video Raster Showing Line Types



The line type acts like a pointer to a data structure containing the information about the “*packet sequence*” for that line type. The packet sequence information is actually held within a pair of registers. For example, the packet sequence for line type 0 is

held in the pair of registers DSI_PKT_SEQ_0_LO and DSI_PKT_SEQ_0_HI. Each pair of packet sequence registers contains all the information required to generate up to 6 packets. This is sufficient to generate any packet sequence for any DSI video line. The information stored for each packet is shown in the following table:

Table 82: Packet Sequence Description Fields

Field	No. bits	Description
PKT_*_SIZE	3	Pointer to packet size register
PKT_*_ID	6	Packet ID as defined in the MIPI DSI spec.
PKT_*_EN	1	Enable. Determines which packets are active.

Using the same pair of registers as an example, Table 83 shows how all six packet descriptors fit into the pair of 32-bit packet sequence registers.

Table 83: Packet Sequence 0 Registers

Register	Field	Bits	Description
DSI_PKT_SEQ_0_LO	PKT_00_SIZE	2:0	Packet 0 Size
	PKT_00_ID	8:3	Packet 0 ID
	PKT_00_EN	9	Packet 0 Enable
	PKT_01_SIZE	12:10	Packet 1 Size
	PKT_01_ID	18:13	Packet 1 ID
	PKT_01_EN	19	Packet 1 Enable
	PKT_02_SIZE	22:20	Packet 2 Size
	PKT_02_ID	28:23	Packet 2 ID
	PKT_02_EN	29	Packet 2 Enable
	SEQ_0_FORCE_LP	30	End in LP state
DSI_PKT_SEQ_0_HI	PKT_03_SIZE	2:0	Packet 3 Size
	PKT_03_ID	8:3	Packet 3 ID
	PKT_03_EN	9	Packet 3 Enable
	PKT_04_SIZE	12:10	Packet 4 Size
	PKT_04_ID	18:13	Packet 4 ID
	PKT_04_EN	19	Packet 4 Enable
	PKT_05_SIZE	22:20	Packet 5 Size
	PKT_05_ID	28:23	Packet 5 ID
	PKT_05_EN	29	Packet 5 Enable

Finally, the SIZE field in the packet descriptor is used as a pointer to a 16-bit "*packet length*" field in the packet length registers. An indirect pointer is used rather than storing the packet length directly in the packet descriptor because the packet lengths are physically large – 16 bits – but there is a lot of re-use. There are only a few different lengths used in a typical raster layout. Thus, it is more efficient to hold the lengths in separate length registers and then to simply refer to them with a short pointer. This allows the storing of 36 packet descriptors in only 6 pairs of packet sequence registers.

Note that one of the packet length registers is reserved for short packets. If a `SIZE` field in the packet descriptor is programmed to 0, this packet is likely a short packet. Short packets are fixed in length to 4 bytes including the packet header byte and ECC byte. In the context of video mode, they are essentially reserved for timing packets. All other packet length registers can be used to determine the length of any associated long packet.

The `SEQ_0_FORCE_LP` bit is used to determine if the link should be placed in the LP state at the end of the packet transmission. In Burst Mode operation, the link always drops back into LP state at the end of the HS packet transmission on a line. However, for Non-Burst Modes, the HS transmission may continue to the next line. It is important the that the hardware

state machine that generates packets not attempt to go to the LP state, but should instead prepare for the next sequence of packets associated with the next video line and keep the line in the HS transmission state.

The packet length registers are shown in Table 84. The packet size pointer in the packet descriptor can point to any of the available length registers – with the one exception of short packets, whose SIZE field must point to length register 0. It is recommended that the suggested packet length assignment for various length registers is followed to reduce confusion when debugging packet descriptors and packet sequences.

Table 84: Packet Length Registers and Assignments

Register	Field	Suggested Assignment
DSI_PKT_LEN_0_1	LENGTH_0	Must be used for short packets
	LENGTH_1	Horizontal sync active length (HSA)
DSI_PKT_LEN_2_3	LENGTH_2	Horizontal back porch length (HBP)
	LENGTH_3	Horizontal active length (ACTIVE)
DSI_PKT_LEN_4_5	LENGTH_4	Horizontal front porch length (HFP)
	LENGTH_5	- reserved -
DSI_PKT_LEN_6_7	LENGTH_6	- reserved -
	LENGTH_7	- reserved -

When programming the length registers, remember that these contain byte counts, not pixel counts. Therefore, the DSI pixel format, number of DSI lanes in use and the finite (non-zero) size of the packet header and CRC words should be taken into account when programming these values – especially for blanking packets since all of these parameters affect the number of bytes that should be used to emulate a specific blanking time. The equations required to determine the byte count in the packets are given in the examples that follow. The identification of the length given in the tables and equations has been included in three video mode examples so that the values in the examples can be easily tied to the registers.

26.4.3 SOL Delay Programming

In order to ensure that the pixel FIFO from display to DSI does not underflow when in video mode, it is necessary to delay the start of packet generation by the DSI by a fixed amount from the arrival of the SOL signal from the display. This is especially true of Burst-Mode, because the DSI byte clock can be very much faster than display pixel clock. If no delay is applied, the DSI will very quickly consume all the pixels in the FIFO and the FIFO will underflow. The SOL_DELAY registers are used to set this delay.

26.4.3.1 Non-Burst Mode

In non-burst mode, the rate at which the DSI consumes pixels is the same as the rate at which the display module produces them, so it is merely sufficient to ensure that enough pixels have been fed into the FIFO to overcome the internal latency. This internal latency is approximately 8 pixel clock cycles. However, SOL_DELAY is programmed in DSI byte clocks, so the pixel format and the number of lanes being used should be taken into account when programming this value. Contact your NVIDIA FAE for details on the relationship between display pixel clock and byte clock. These ratios are then used to determine the value of SOL_DELAY as follows:

SOL_DELAY = (((Sol2VldDly) + 12) * F_{DSI} / Fpixel) + FifoLatency

FifoLatency = Ceil (2 * (F_{DSI} / Fpixel)) + 6

Sol2VldDly = Hsync start to Pixel valid delay

Where F_{DSI} and F_{pixel} are the DSI byte and pixel clocks, respectively.

Alternatively, a fixed value of SOL_DELAY that will work for all pixel formats and numbers of lanes can be programming by taking the worst-case ratio of 1:3 (pixel:byte) clock and multiplying by 8 to give SOL_DELAY = 24.

26.4.3.2 Burst Mode

In Burst mode, not only must the pixel format, number of lanes, and the DSI / pixel clock ratio be taken into account, but also the horizontal back porch time (including Hsync) and the Horizontal active time. So, for Burst mode:

- When EOTp packets are enabled:

  SOL_DELAY = (((Sol2VldDly + 4 + Tactive + 1) * FDSI / Fpixel) –

  ((Tactive + 1) * FDSI_NB / Fpixel_NB))) + FifoLatency

- When EOTp packets are not enabled:

  SOL_DELAY = (((Sol2VldDly + 4 + Tactive) * FDSI / Fpixel) –

  ((Tactive) * FDSI_NB / Fpixel_NB))) + FifoLatency

That is,

FifoLatency = Ceil (2 * (FDSI / Fpixel)) + 6

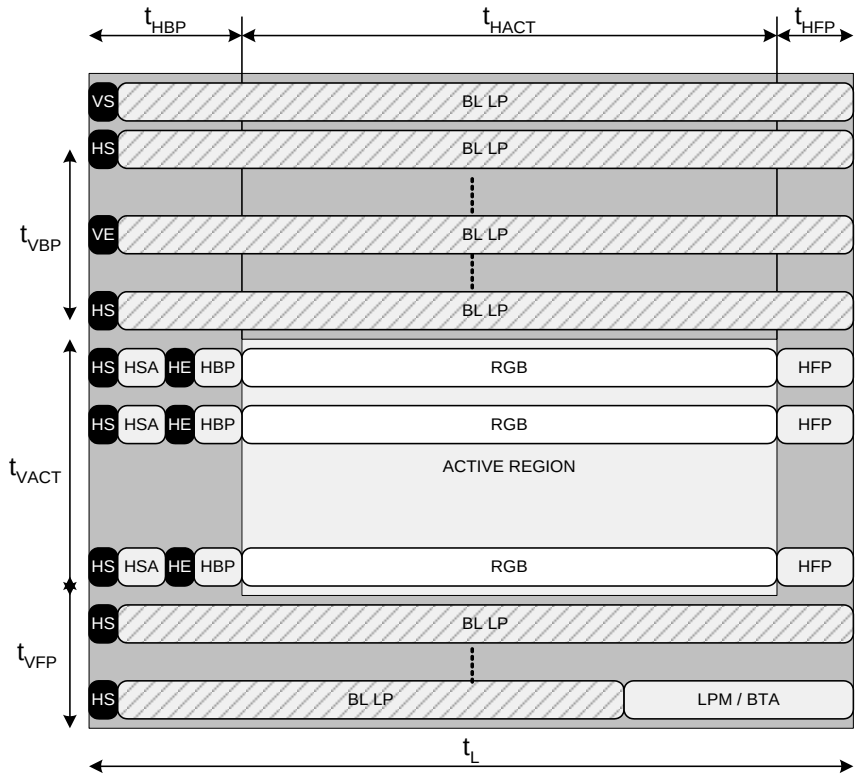
Sol2VldDly = Hsync start to Pixel valid delay

Where the ratio F_{DSI_NB} / F_{pixel_NB} is determined by the clock ratio tables for Non-Burst mode, according to the pixel format used and the number of active lanes.

26.4.4 Non-Burst Mode with Start and End

This mode conveys traditional raster synchronization information across the link to the peripheral by sending both start and end sync packets.

Figure 110: Timing Diagram for Video Non-Burst Mode with Start and End



During the Vertical Active period, all packets should be concatenated into a single HS transmission for the duration of the Vertical Active period.

The figure below shows how a single line is constructed from multiple packets.

Figure 111: Non-Burst Mode with Start and End Packet Timing Detail

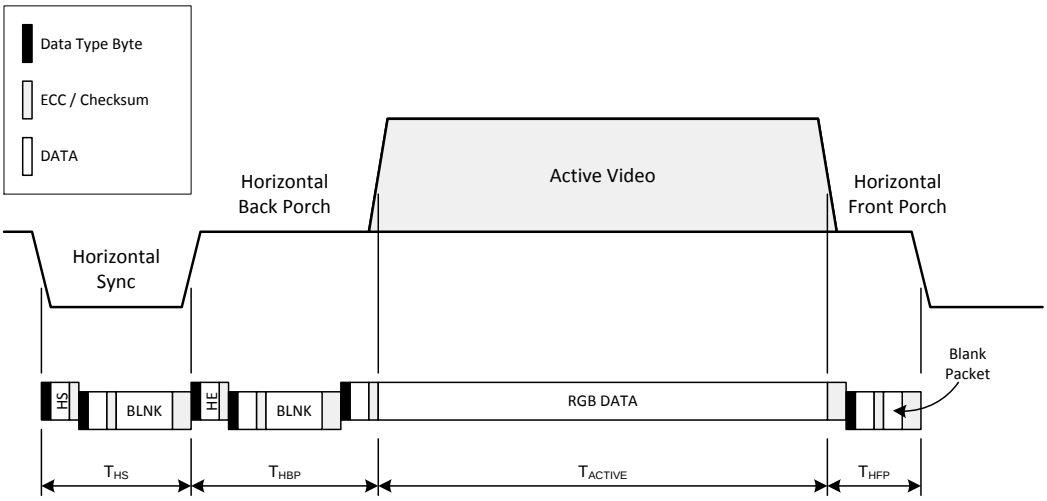


Table 85: Payload Size Table - Non-Burst Mode with Start and End

Packet	Payload Size (Bytes)
HSA	(T_{HS} * B) – 10
HBP	(T_{HBP} * B) – 14
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 8

B = 2, 2.25 or 3, depending on pixel format.

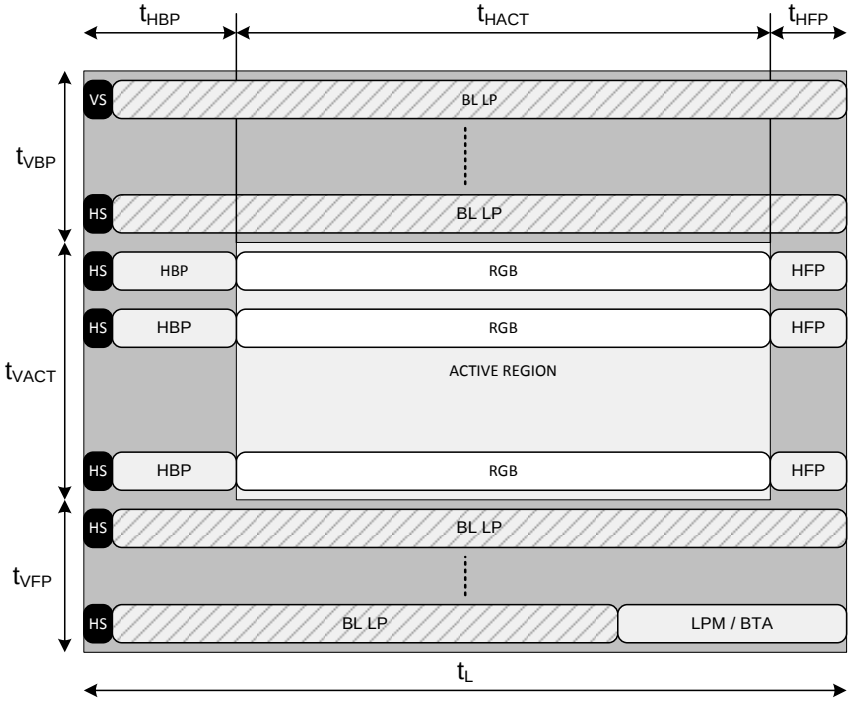
Table 86: Line Type Packet Sequences - Non-Burst with Sync Ends

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	7								
1	VE	0	EOT	7								
2	HS	0	EOT	7								
3	HS	0	BLNK	1	HE	0	BLNK	2	RGB	3	BLNK	4
4	HS	0	EOT	7								
5	HS	0	BLNK	1	HE	0	BLNK	2	RGB	3	BLNK	4

26.4.5 Non-Burst Mode (without Ends)

This mode relaxes the requirement to mimic the generation of sync pulses and merely mandates that the start of the line is indicated and that the pixels appear at the same rate and in the same area of the raster as a tradition raster structure.

Figure 112: Timing Diagram for Video Non-Burst Mode



During the Vertical Active period, all packets should be concatenated into a single HS transmission for the duration of the Vertical Active period.

The figure below shows how a single line is constructed from multiple packets.

Figure 113: Non-Burst Mode Packet Timing Detail

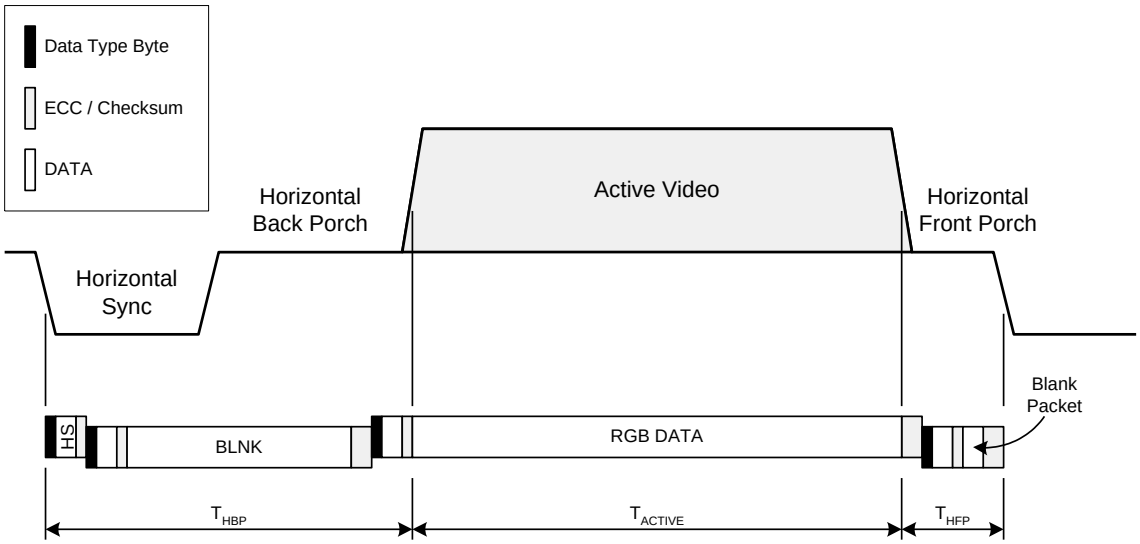


Table 87: Payload Size Table - Non-Burst Mode

Packet	Payload Size (Bytes)
HBP	(T_{HBP} * B) – 14
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 8

B = 2, 2.25, or 3, depending on pixel format.

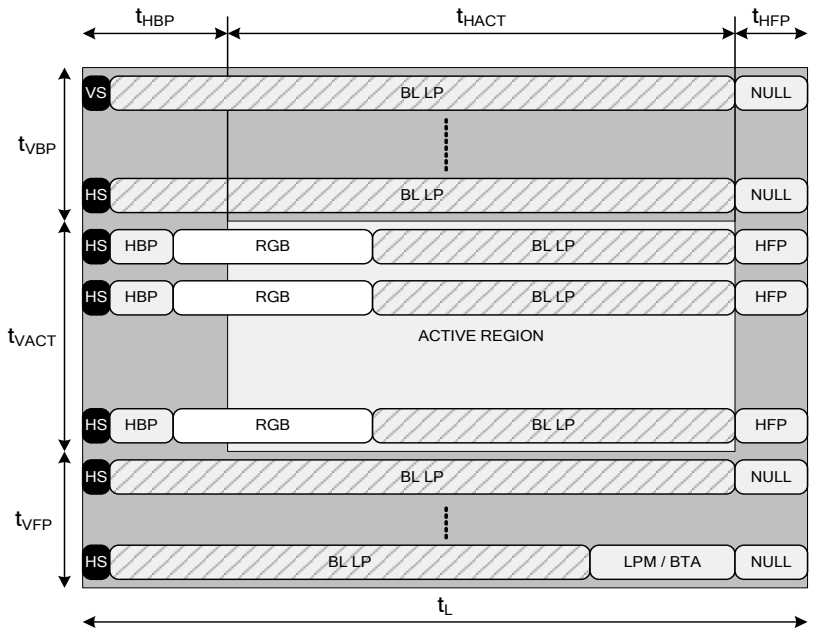
Table 88: Line Type Packet Sequences – Non Burst

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	7								
1	HS	0	EOT	7								
2	HS	0	EOT	7								
3	HS	0	BLNK	2	RGB	3	BLNK	4				
4	HS	0	EOT	7								
5	HS	0	BLNK	2	RGB	3	BLNK	4				

26.4.6 Burst Mode

In Burst Mode, the only attempt to match the raster structure is with the timing of the sync start events. The actual RGB pixel data is transmitted at whatever rate is convenient. This means that the HS, HBP, and RGB packets become compressed with respect to the timing of the underlying raster. This allows some period of idle time for each line that can be used for the transmission of other packets.

Figure 114: Timing Diagram for Video Burst Mode



During the Vertical Active period, packets on an individual line should be concatenated into a single HS transmission. However, unlike Non-Burst Mode, the bus should go idle – if possible – at the end of this transmission. This will allow additional non-video packets to be sent simultaneously with the video stream. The NULL and HFP packets may be optional. Check the latest MIPI DSI specification for clarification.

Table 89: Line Type Packet Sequences - Burst Mode

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	7						
1	HS	0	EOT	7						
2	HS	0	EOT	7						
3	HS	0	BLNK	2	RGB	3	EOT	7		
4	HS	0	EOT	7						
5	HS	0	BLNK	2	RGB	3	EOT	7		

Notes:

- Burst mode always forces the LP packet field to LP mode. DSI_DSI_PKT_SEQ_0_LO/HI_0/1/2/3/4/5[SEQ_0_FORCE_LP] = 1'b1
- The register programming updates in video mode that impact the raster timing (DSI_DSI_PKT_SEQ_*_LO/HI_*, video mode control fields DSI_DSI_CONTROL_0) are assumed to be static with respect to the display controller. On-the-fly updates are not supported.

This programming sequence must be followed when switching the modes/updates to raster timing:

26. Disable the DSI (DSI_LEG_EN disable)
27. Disable the DC2DSI interface path (DC_DISP_DISP_WIN_OPTIONS [DSI_ENABLE] = 1'b0)
28. Update the DSI configuration registers.
29. Enable the DSI (DSI_LEG_EN enable)
30. Enable the DC2DSI interface path (DC_DISP_DISP_WIN_OPTIONS [DSI_ENABLE] = 1'b1).

26.4.7 Sequence while Switching the Modes/Updates to Raster Timing

The following programming sequence need to be followed while switching the modes/ updates to raster timing.

1. Disable DC2DSI interface path (DC_DISP_DISP_WIN_OPTIONS [DSI_ENABLE] = 1'b0).
2. Disable DSI (DSI_LEG_EN disable).
3. Update the DSI configuration registers.
4. Enable the DC2DSI interface path (DC_DISP_DISP_WIN_OPTIONS [DSI_ENABLE] = 1'b1).
5. Enable DSI (DSI_LEG_EN enable).

26.4.8 Command Mode Programming

There are two sources of command mode packet sequences:

- Display Controller
- Host Interface

Only one of these sources will be active at any particular time.

26.4.8.1 Command Mode from Host

In this mode of operation, all packets sent over the DSI interface are determined by software. There may be hardware assistance in the generation of error correction codes (ECCs) and cyclic redundancy checks (CRCs), but all other data is passed unaltered to the DSI physical layer.

Host Packet Writes

Packets are written to the hardware by writing to the DSI_WR_DATA register. The data written is passed into the DSI Host transmit FIFO. If the data is a packet header and the ECC_ENABLE field in the HOST_DSI_CONTROL register is set to

ENABLE, then the MSBs of the 32-bit packet header word are replaced with a hardware-computed ECC byte prior to being written into the FIFO. If this field is set to DISABLE, then no action is taken by the hardware and the packet header is written unchanged.

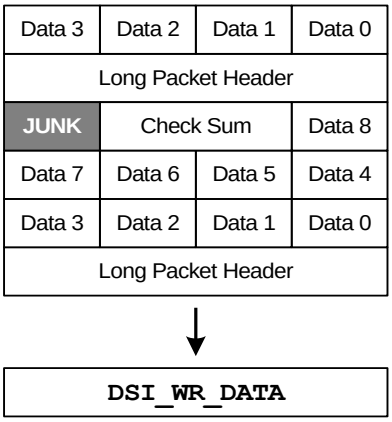
If the packet is a long packet, then the packet payload should be written to the register after the packet header is written. If the ECC_ENABLE field of the HOST_DSI_CONTROL register is set to ENABLE, then the hardware computes the check-sum and appends it to the packet information. If this field is set to DISABLE, then software must append the correctly computed check-sum to the packet data.

Rules:

1. Software should make sure packets are transmitted in their entirety and that once transmission starts, the FIFO contains enough data to finish a transition on a packet boundary. This can be achieved by only initiating a transmission – either explicitly or indirectly – once all data required for a transmission has been written to the transmit FIFO.
2. In Type-1 Display modules (i.e., the display device includes the full frame buffer to hold image data), the complete image data can be written to the frame buffer via the Host command mode. Software configures the register field to enable the Frame buffer selection for transfer of high-resolution image data.
   DSI_HOST_DSI_CONTROL[PKT_WR_FIFO_SEL]

   Note: The maximum length of the long packet supported is 1920 words including header, payload, and CRC.
3. Packets should be written such that the packet header is always written in one 32-bit word and is never split across writes. In other words, if the end of a packet does not fall on a 32-bit word boundary (if the payload has an odd length, for example), then the header for the next packet should not border the last packet, but should realign with the register. See Figure 115 for an example.

Figure 115: Packet Alignment for Writes to DSI_WR_DATA



Note: For unaligned word transfers, the host transactions are split into multiple host transfers at the unaligned boundary.

Host Packet Transmission

When the Host write FIFO starts to drain (flush) is programmable via an explicit FLUSH bit in a control register

Immediate (HOST_TX_TRIG_SRC == IMMEDIATE**)**

As the name implies – if a '1' is written to the `DSI_HOST_TRIGGER` field of the `DSI_TRIGGER` register, then transmission of the data in the Host write FIFO will start immediately. It is recommended that all data required for transmission is written to the DSI_WR_DATA register prior to setting this bit.

26.4.8.2 Command Mode from Display Controller

This mode of operation allows the display controller to write pixel data packets to a DBI-like device that does not require data to be sent in an isochronous raster structure like a DPI device. The data coming from the display controller is sent in an isochronous manner, but the commands sent will be DCS control commands, rather than Video Mode timing and blanking packets.

Setup

There are six steps to operating in this mode:

1. Program the Peripheral display device using DCS commands via the Host Command interface. It is important to send the set_column_address, set_page_address, and set_pixel_format commands. These effectively define the area to which you will be writing pixels.
2. Program the DSI_INIT_SEQ_CONTROL and DSI_INIT_SEQ_DATA_* registers in the DSI interface with the appropriate values. Any commands required to be sent once per frame and *every* frame by way of setup prior to the pixel data being sent should be put in here.
3. Program the `DSI_DATA_FORMAT` field of the DSI_CONTROL register to the required pixel format. Note that `BIT18NP` should not be programmed – see the section below on pixel format restrictions.
4. Program the DSI_PKT_SEQ_* registers. Program the registers in the following way:
   - Packet Sequence registers for Line Types 0, 1, 2, and 4 should all be programmed with 0 in all the Packet Enable fields. In other words, there should not be any packets generated for these line types.
   - Packet Sequence registers for Line Types 3 and 5 should be programmed with a DCS Long Write packet.
5. Program the DCS command ID register to have a write_start command associated with Line Type 5 (First line of active) and a write_continue command associated with Line Type 3 (all other active lines).
6. Enable the display. When Vertical syncs arrive from the display, the initialization sequence should be sent, and for every active line, the pixel data will be sent in a DCS Long Write packet.

Pixel Format Restrictions

The MIPI DCS specification allows for up to 6 different pixel data formats to be transmitted in a write_start or write_continue command. These pixel formats are listed in Table 90. The DSI supports 3 of these pixel formats.

The formats supported are also shown in the “supported” column of Table 90. Do not set `BIT18NP` as a pixel format. There is no DCS equivalent of this format, so undefined behavior may result.

Table 90: DCS Pixel Format Support

DCS ID	DCS Format	Supported	Name
0	reserved	N/A	-
1	3bpp	N	-
2	8bpp	N	-
3	12bpp	N	-
4	reserved	N/A	-
5	16bpp	Y	BIT16P
6	18bpp	Y	BIT18P
7	24bpp	Y	BIT24P

Simultaneous Host Command Packets

It is necessary, from time to time, to send a DCS command to the display peripheral in a “side-band” fashion while the display controller is continuing to send DCS write commands with pixel information. Line Type 4 (first blank line) is reserved to indicate to the hardware when it should attempt to send any DCS command packets requested by software.

If it is desired to send a DCS command in this way, the DCS command packet should be written in its entirety (header, ECC, DCS command, payload, CSC) into the Host Command FIFO. The HOST_TX_TRIG_SRC field should then be set to IMMEDIATE.

The DSI hardware will then send the entire contents of the FIFO out on the DSI interface on the first line of blanking. Sending the data on this line guarantees the Host packet transmission will have enough time to complete before the next packet generated by the Display Controller is generated.

> **Note:** There is no BTA permitted in this mode, so no ACK, ACK with error report, or read return data will be generated.

If the host transfer is enabled during the frame video blanking interval, the following sequence needs to be performed at the end of the frame:

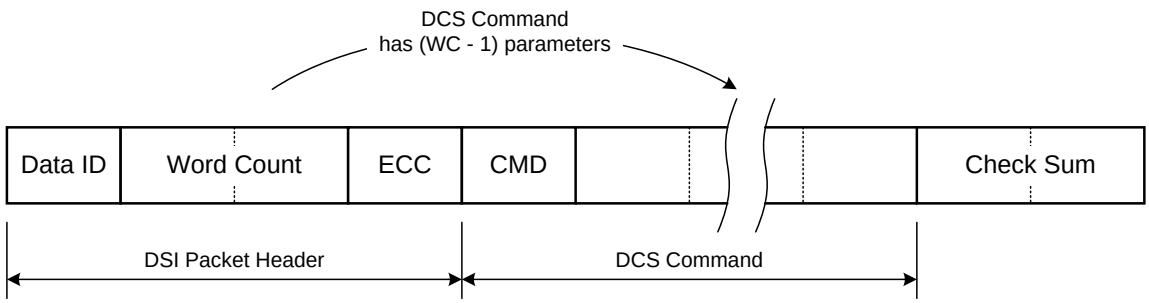
1. Wait for an end of frame event from the display controller.
2. Set the increment syncpt soft reset; that is, program the INCR_SYNCPT_SOFT_RESET field in the DSI_INCR_SYNCPT_CNTRL_0 register to 1’b1.
3. Clear the increment syncpt soft reset; that is, program the INCR_SYNCPT_SOFT_RESET field in the DSI_INCR_SYNCPT_CNTRL_0 register to 1’b0.
4. Continue with the subsequent configuration.

26.4.9 Display Command Set (DCS) Packets

DCS is a legacy control command set that is used to program various control registers present in typical LCD panels used in cell phones. Historically, the commands would be sent to the display over a MIPI DBI interface. Since MIPI DSI is meant to replace both DPI and DBI, it must also transport these control commands.

There are several DCS-specific commands in the MIPI DSI specification that can be used to send DCS commands to the Peripheral Display Device. However, the most useful is probably the DCS Long Write packet since it can be used to send commands with more than one parameter. The DCS command is embedded with an MIPI DSI Long Packet as follows: The DCS Command Byte and all the DCS Command Parameters (payload) are concatenated and sent in the Payload of the MIPI DSI Long Packet. The Word Count of the DSI packet conveys the total size of the DSC packet. As usual, a 2-byte CS footer is appended to the DSI Long Packet. The figure below shows how a DCS Long Write packet is constructed.

Figure 116: DCS Command Placement in DSI Long Packet



16 bpp Byte Ordering in 16 bpp Format for Video/Command Mode

The DSI specification suggests a different byte order in the 16 bpp video and command mode data bit order, i.e., data packing in:

Video mode [16:0] -> {B[4-0],G[5-3]},{G[2-0],R[4-0]} (Default)

Command mode [16:0] -> {G[2-0],B [4-0]},{R[4-0],G[5-3]} (SwapEn)

The byte order can be configured through the DSI_DSI_CONTROL_0 register, DFMT_16BPP_SWAP_EN field. By default, the video mode order is selected and the command mode can be selected by the programing DFMT_16BPP_SWAP_EN to 1.

Frame Synchronization Between Identical Command Mode Displays

When stretching a single surface across multiple identical displays, frame synchronization can be achieved to avoid visual artifacts due to timing drifts across both the displays.

DCS command “adjust_vsync_timing” provides a mechanism for adjusting the command mode display vsync/frame sync timing with respect to another identical display. Vsync can be shifted, i.e., delayed/advanced by the specified number of scan lines.

Host Command Mode Byte Transmission

Command mode operation primarily involves sending the commands and data to the peripheral (the display device). The host processor indirectly controls activity at the peripheral by sending commands, parameters, and data to the display controller. Command Mode operation requires a bidirectional interface capabilities.

Command mode transfers are classified into,

- HS/LP mode

  Host command mode transmission can be enabled in High Speed and Low Power modes. Follow this programming sequence to configure the DSI to enable command mode transactions.

  - Configure DSI_DSI_CONTROL_0 [DSI_HOST_ENABLE]
  - Enable checksum/Ecc mode via DSI_HOST_DSI_CONTROL_0[CS_ENABLE], …, DSI_HOST_DSI_CONTROL_0[ECC_ENABLE].
  - Write the host data to the register DSI_DSI_WR_DATA_0. Once all the command packets are pushed into the FIFO, trigger the start of the transaction via the HOST_TX_TRIG_SRC field in the DSI_HOST_DSI_CONTROL_0 register.
  - LP/HS mode of transmission is selected by configuring the DSI_HIGH_SPEED_TRANS in the DSI_HOST_DSI_CONTROL_0 register.

- Raw Mode

  Host command mode transmission also supports RAW data bytes to be transferred to the display device. In Raw mode, the data written to the host is transferred with no concept of packets and the DSI hardware does not decode the packet headers and ECC/CS computation is not performed.

  In order to send the RAW data bytes (debug / diagnostic workaround mode):

  - Program the number of bytes to be transferred to the DSI_RAW_DATA_BYTE_COUNT register. The byte count programmed is equal to the total number of raw bytes (not to be interpreted as the byte count of a packet).
  - Write raw data bytes to the DSI Host.
  - Enable RAW_DATA in the DSI_HOST_DSI_CONTROL_0 register.

26.4.10 High-Speed Clock Configurations

In High Speed mode, the Host provides a low-swing, differential DDR clock signal for high-speed data transfers. The high-speed clock lane is configurable via the DSI_HS_CLK_CTRL] field in the DSI_DSI_CONTROL register.

- Free Running Mode /Continuous

  The D-Phy clock lane operates in HS free running or continuous mode; i.e., used in systems where the D-Phy clock lane acts as a clock source to the display devices eliminating the need for an alternate oscillator clock source.

- TX Mode/Discontinuous Mode

  The D-Phy clock lane operates in burst mode or discontinuous mode; i.e., the D-Phy clock lane remains active only during the HS transmission and stops the clock lane when there are no high-speed transmissions active in any of the data lanes.

Notes:

1. The Host programs the total horizontal blank period to meet the following criteria to provide enough time for HS <-> LP transitions.

   (HBLANK) >= (number_of_lanes * (HS_Trail + HS-Exit + CLK_POST + CLK_TRAIL + HS_EXIT + TLPX + TLPX + TCLK_PREPARE + TCLK_ZERO + TCLK_PRE + 1 + TLPX + THS_PREPARE + TDAT_ZERO + 1))

   The parameters in this equation are defined in the DSI_DSI_PHY_TIMING registers.

2. The DSI_HS_CLK_CTRL field in the DSI_DSI_CONTROL_0 register control field for the HS clock lane is programmed when the DSI_HIGH_SPEED_TRANS control field in the DSI_DSI_CONTROL_0 register is programmed for HS packet transmission of packets.
3. The HS clock lane state transitions from continuous to discontinuous with soft reset assertion.

26.4.11 Ultra-Low Power Sequence

In this procedure, starting from the Stop state, the transmit side drives the TX-ULPS-Rqst state (LP-10) and then drives the TX-ULPS State (LP-00). After this, the clock lane enters the Ultra-Low Power state. If an error occurs, and an LP-01 or LP-11 is detected immediately after the TX-ULPS-Rqst state, the Ultra-Low Power state entry procedure is aborted, and the receive side waits for or returns to the Stop state, respectively.

The Clock lane and Data lanes can be configured to enter and exit ULPS independently/simultaneously:

- Independent control of the ULPS

  The Clock needs to be in the HS state, sending the active clock when data lanes are entering ULPS, When entering ULPS, the data lanes enter the ULPS first, followed by the clock lanes. Similarly, when exiting ULPS, the Data lanes exit the ULPS first, followed by the clock lanes.

  DSI_ULTRA_LOW_POWER_CONTROL register configures the control to enter the ULPS on particular clock/Data lane.

  ULPS Entry

  - Program the data lane to enter ULP sequence, wait for syncpt.
  - Program the clock lane to enter ULP sequence and, wait for syncpt

  ULPS Exit

  - Program the data lanes to exit the sequence, wait for syncpt
  - Program the clock lanes to exit the sequence, wait for syncpt

- Simultaneous control of the ULPS

  The DSI_ULTRA_LOW_POWER field in the HOST_DSI_CONTROL register controls the ULPS for all lanes (clock and data lanes 0-3) simultaneously.

Notes:

- The DSI_TWAKEUP field in the DSI_PHY_TIMING_2 register defines the length of the exit interval in multiples of 512 byte clocks common for all lanes (Clock/Data).
- The ULPM entry sequence always starts from the NORMAL state. Software should place the control fields in the NORMAL state after the exit sequence after the last operation is done. That is, in case of a ULPM sequence on clock, data lanes, you are recommended to program the register to the NORMAL state after the data lane exit sequence.

26.4.12 End of Transmission Packet (EOTp)

The DSI specification defines a dedicated End of Transmission packet (EoTp) at the protocol layer to indicate the end of HS transmission. For backwards compatibility with earlier DSI systems, this EoTp can be enabled or disabled.

For host transactions during the video mode, EOTp is programmed in the Packet Sequence register, and host data is sent on the interface followed by the last video packet byte, including the EOTp packet as the last word. That is, software programs the EOTP packet as part of the host data during host transactions during video mode.

26.4.13 Host Command Packet During Video Mode Transmission

DSI supports transmission of host command packets during vertical blanking in Video mode.

Host packet transmission during vertical blank time is possible and programmable through the DSI_VID_MODE_CONTROL register to enable and select the line type to trigger the host command packets during video mode. The 0/1/2/4 line types with single short packet transmission as part of video raster can enable simultaneous host transfer in Video mode.

Video blank line types are:

- Line Type = 0 - Blank line starting with VS
- Line Type = 1 - Blank line starting with VE
- Line Type = 2 - Blank line starting with HS
- Line Type = 4 - First Blank line after active starting with HS.

The programming sequence to enable host transactions during Video mode is:

1. Choose the line type 0/1/2/4 (Video Blanking) for simultaneous host transfers during Video mode. Program the DSI_LINE_TYPE field in the DSI_DSI_VID_MODE_CONTROL_0 register to FOUR.
2. Simultaneous host transfers are enabled during Video mode via the Video mode control field. Program the DSI_CMD_PKT_VID_ENABLE field in the DSI_DSI_VID_MODE_CONTROL_0 to 1’b1.
3. Write host data transactions to the host data buffer via the DSI_DSI_WR_DATA_0 register.

HS Packet	Host Data Packets (<= 64 Words)

4. Trigger host transmission when the complete host data for transmission is pushed into the buffer. Program the DSI_HOST_TRIGGER field in the DSI_DSI_TRIGGER_0 to 1’b1.
5. Wait for FRAME_DONE syncpt from the display controller.
6. Set the increment syncpt soft reset; that is, program the INCR_SYNCPT_SOFT_RESET field in the DSI_INCR_SYNCPT_CNTRL_0 register to 1’b1.
7. Clear the increment syncpt soft reset; that is, program the INCR_SYNCPT_SOFT_RESET field in the DSI_INCR_SYNCPT_CNTRL_0 register to 1’b0.

Notes:

- The host packet data transmission cannot be enabled in Video burst mode, where NULL_PACKETs are included with the HSYNC packet.
- Checksum/ECC computation for the host packets can be performed by either the hardware or software by programming the CS/ECC_Enable fields in the DSI_HOST_DSI_CONTROL_0 register.
- After a single VSYNC/VSYNCEND/HSYNC short packet is transmitted, if the Host transmission is enabled, software-constructed Host data packets are transmitted.
- The length of the Host FIFO data is limited to 64 words deep.
- Host data should be pushed into the Host FIFO before the frame is triggered.
- Host packets must be aligned to a word boundary. For host packets ending on an unaligned word boundary, there is support for a maximum of 4 such consecutive packets transferring in a worst-case configuration scenario and relaxed for other configurations. In the worst-case configuration scenario, the number of lanes is 4, and the Host packets end on Byte 0 (that is, a single byte is written in the last word).

  For an unaligned word boundary, if the end of a packet does not fall on a 32-bit word boundary, then the header for the next packet should not border the last packet (see the figure showing packet alignment for writes to DSI_WR_DATA).

26.4.14 Ganged Mode (Odd-Even/Left-Right) Programming

The reader is expected to be familiar with the DSI specification and normal video mode programming guidelines before reading this section on Ganged Mode.

Odd-Even

For Odd-Even Ganged mode, the programming guidelines are based on these assumptions/limitations:

1. Both partitions have a single PLLD and clock source (DSI instances)
2. For VNB, all data lanes end in the same clock cycle aligned to the number of configured lanes.
3. The number of pixels in the 18 bpp packed case is always a multiple of four.
4. The number of active pixels from the display controller is not greater than 4096 pixels.

Programming guidelines/sequence and equations for Odd-Even Ganged mode are listed below:

1. Required inputs are: image resolution data format, and optional number of lanes
2. Compute the pixel clock frequency from the given resolution and specification
   pixel frequency = (TotalHorizontalPeriod * TotalVerticalPeriod * FrameRate) / 10^6
3. Determine the possible shift clock divider value and ganged mode image split information from the number of pixels, data format, and pixel frequency

   (pixel frequency = byte frequency * 4/clock divider)
4. Symmetrical split is possible for all resolutions. Determine the possibility of an asymmetrical split.

   Let n1 = number of lanes for the first partition and n2 = number of lanes for the second partition from step 3.

   totalNumLanesInGangedMode = n1 + n2
5. Determine the correction pixels needed to align the totalNumPixels and totalNumBytes with totalNumLanesInGangedMode.
   - The pixelsNeededToInitiallyAlign should be such that the updated totalNumPixels is a multiple of LCM (total Num Lane sInGangedMode,X).
   - totalNumBytes = totalNumPixels*bytesPerPixel is a multiple of totalNumLanesInGangedMode

Notes:

- LCM – least common multiple (X=4 for 18bpp packed else X=1)
- In 18bpp packed case, else X=1. If remainder is non-zero/ganged alignment was not successful (from STEP 9), add 'n' number of additional active pixels (zeros/random data) so that the updated image active width is divisible by the LCM above. This is needed to support image split as determined in step 4.
- Add the additional correction/dummy pixels to HFP, determine new pixel frequency and compute the byte frequency using the clock divider. Use this information to program the PLLD and shift clock divider. It is okay to add dummy active pixels since panels provide an option using a register field that can be configured to drop/discard the padded pixels.
- Update total number of pixels now with the newly added active pixels, ensure the total number of active pixels does not exceed 4096 pixels. In case the total number of pixels exceed 4096 pixels then and the frame is considered not to support asymmetrically split for the given configuration, i.e.,,frame rate and split configurations.
- In the 18 bpp packed format case, you cannot complete more than a definite set of iterations for a given frame rate. If the alignment is not achieved in within 50-60 pixels, we assume the pixel alignment is not possible for the frame for asymmetric split for that frame rate.

6. Determine if the (first partition width) : (second partition width) ratio = n1/n2. If successful, proceed to the next step. If not, ganged width alignment is not successful. Repeat from step 5 by adding more active pixels that still meet the requirement mentioned there.

splitFactor (for first partition)= n1/ totalNumLanesInGangedMode;
First Partition Width (active pixels)= CEIL(spliFactor*image width)
Pixels per line of first Partition = CEIL(splitFactor*totalNumPixels)
second partition active width = imagewidth – first partition width
pixels per line of second partition = totalNumPixels - Pixels per line of first Partition

Note:

- In consideration of constraint #4, software will ensure that the total number of pixels does not exceed 4096 pixels. If the total number of pixels exceeds 4096 pixels, then the frame does not support asymmetrically split for the given configuration, i.e., frame rate and split configurations.
- The number of active pixels added is limited to no more than 60 pixels. If this value exceeds 60 in these iterations, the input image cannot be split as per the split ratio (cannot deviate much from the selected frame rate).

7. Determine the new pixel frequency (new active pixels and HFP correction) and compute the byte frequency using the clock divider. Use this information to program the PLLD and shift clock divider.
8. Determine the factors of the first partition width and second partition width.

   Select 'x' from the factors of the first partition and 'y' from the factors of the second partition such that x:y = n1/n2. The minimum of 1:1 (symmetrical) is possible for any image.

   To send the first group of pixels, ganged start pointer = 0, valid width = x, and low width = y. For the second group, ganged start pointer = x, valid width = y, and low width = x.
9. In Odd-Even Ganged mode, program the HSYNC, HACT, and HFP packets.

   HACT payload size = Total Actual Image Bytes on Data lanes

   Initial HFP payload size = totalNumBytesForTheSelectedPartition - numActBytesToBeSent - 16 (16 is because of 4 Bytes - HSYNC, (4+2) Bytes - HACT, (4+2) Bytes - HFP)

10. Compute the correction bytes (per line of partition) again to align with the line width and number of lanes (per partition), limitation #2

    correctionBytesPerLane = CEIL(totalNoOfPixelsPer HorzLine*(FDSI/ Fpixel) – (BPP/ Gangedmode_lanes))

    totalNumBytesForTheSelectedPartition = (correctionBytesPerLane * numOfLanesForSelectedPartition)

    total Horizontal Blank(Bytes) = (totalNumBytesForTheSelectedPartition -16+ correctionBytesForAlignment)

11. Ganged mode register programming
    - Even Partition

      DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1;

      DSI_DSI_GANGED_MODE_START_0[DSI_GANGED_START_POINTER] = 0

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_HIGH_WIDTH] = Even partition active width

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_LOW_WIDTH] = Even partition inactive/low width.
    - Odd Partition

      DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1;

      DSI_DSI_GANGED_MODE_START_0[DSI_GANGED_START_POINTER] = High width of Even partition

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_HIGH_WIDTH] = odd partition active width.

      DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_LOW_WIDTH] = odd partition inactive/low width.

12. SOL delay calculation:

    solFactor = ((Sol2VldDly + 6) * (FDSI / Fpixe)) + 6;

    sol_delay =

    Ceil (TotalHorzPixelWidth * (FDSI / Fpixel)) -
    Ceil (((SplitFactor * TotalHorzPixelWidth) * BPP) /
      Gangedmode_lanes) + SolFactor;

    Final_sol_delay = (dcs_mode) ? (sol_delay + 20) : (sol_delay)

    DSI_DSI_SOL_DELAY_0 = Final_sol_delay

    Notes:
    - Sol2VldDly = Hsync start to Pixel valid delay
    - totalNoOfPixelsPerHorzLine = Total Number of horizontal pixel count.
    - SplitFactor = DSI_partition_lanes / Gangedmode_lanes.
    - Gangedmode_lanes = Total number of lanes required in Ganged mode transfer
    - DSI_partition_lanes = Total lanes in current partition/channel.
    - BPP = Bytes per pixel

Left-Right:

For Left-Right Ganged mode, the programming guidelines are based on these assumptions/limitations:

1. Both partitions have a single PLLD and clock source (DSI instances)

2. For VNB, all data lanes end in the same clock cycle aligned to the number of configured lanes (per the DSI specification).
3. The number of pixels in the 18 bpp packed case is always aligned to 4 pixels (per DSI specification).
4. The number of active pixels from the display controller is not greater than 4096 pixels.

Programming guidelines/sequence and equations for Left-Right Ganged mode are listed below:

1. Required inputs are Image resolution and data format, and optional number of lanes
2. Compute the pixel clock frequency from the given resolution and specification
   pixel frequency = (TotalHorizontalPeriod * TotalVerticalPeriod * FrameRate) / 10^6
3. Determine the shift clock divider value and ganged mode image split information from the number of pixels, pixel frequency, and data format.

   (pixel frequency = DSI clock (byte frequency) * 4/clock divider)
4. Symmetrical split is possible for all resolutions. Determine the possibility of the asymmetrical split.
5. Let n1=number of lanes for first partition and n2 = number of lanes for second partition from step 3

   totalNumLanesInGangedMode = n1 + n2
6. Determine the correction/additional pixels needed to align the totalNumPixels and totalNumBytes with totalNumLanesInGangedMode.
   From Limitation #3:

   The pixelsNeededToInitiallyAlign should be such that updated totalNumPixels is a multiple of LCM (total Num Lane sInGangedMode,X)

   totalNumBytes = totalNumPixels*bytesPerPixel

   Ensure the total Number of Bytes to be a multiple of totalNumLanesInGangedMode.

   Note: For the least common multiple (LCM), X = 4 for 18 bpp packed; otherwise X = 1

   In the 18 bpp packed format case, you may not complete over a definite set of iterations for a given frame rate. Hence check if the total additional pixels added to Align doesn't exceed 50-60 pixels. If it exceeds which means pixel alignment not possible and the frame considered not to support asymmetrically split for the 18BPP packed case (for that frame rate)..

   Add the additional correction pixels to the HFP, determine a new pixel frequency, and compute the byte frequency using the clock divider. Use this information to program the PLLD and shift clock divider.
7. Determine the number of pixels per split on the computed Pixel count from step 6:
   - Determine the first partition width, and then the remaining pixels will be in the second partition.

   splitFactor (for first partition)= n1/ totalNumLanesInGangedMode

   First Partition Width (active pixels)= CEIL(splitFactor*image width)

   Pixels per line of the first partition = CEIL(splitFactor*totalNumPixels)

   second partition active width = imagewidth – first partition width

   pixels per line of the second partition = totalNumPixels - Pixels per line of the first partition.

For 18 bpp packed, software should recheck if each of the widths determined above is multiple of 4. Otherwise, adjust the active pixels of the partition (increase) to align to 4 pixels and correspondingly decrease the pixels in the HFP (to still meet the line time) accordingly. Adjust pixels in the second partition, too.

numActBytesToBeSent = partition active width * bytesPerPixel

totalNumBytesForTheSelectedPartition = pixels per line of selected partition * bytesPerPixel

8. In Left-Right Ganged mode, program the HSYNC, HACT, and HFP packets.

   HACT payload size = Total Actual Image Bytes on Data lanes

   Initial HFP payload size = totalNumBytesForTheSelectedPartition - numActBytesToBeSent - 16

   (16 is because of 4 bytes - HSYNC, (4+2) Bytes - HACT, (4+2) Bytes - HFP)

9. Compute the correction bytes (per line of partition) again to align with line width and number of lanes (per partition) (limitation #2)

   correctionBytesPerLane = CEIL(totalNoOfPixelsPerHorzLine *(FDSI/ Fpixel) – (BPP/ Gangedmode_lanes))

   totalNumBytesForTheSelectedPartition = (correctionBytesPerLane * numOfLanesForSelectedPartition)

   total Horizontal Blank(Bytes) = (totalNumBytesForTheSelectedPartition -16+ correctionBytesForAlignment)

10. Ganged mode register programming
    a. Left Partition

       DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1;

       DSI_DSI_GANGED_MODE_START_0[DSI_GANGED_START_POINTER] = 0;

       DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_HIGH_WIDTH] = Total horizontal active width

       DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_LOW_WIDTH] = Total horizontal width – Total horizontal active width.

    b. Right Partition

       DSI_DSI_GANGED_MODE_CONTROL_0[DSI_GANGED_MODE_EN] = 1;

       DSI_DSI_GANGED_MODE_START_0[DSI_GANGED_START_POINTER] = leftPartitionWidth (for right partition)

       DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_HIGH_WIDTH] = Total horizontal active width

       DSI_DSI_GANGED_MODE_SIZE_0[DSI_GANGED_VALID_LOW_WIDTH] = Total horizontal width – Total horizontal active width.

11. SOL delay calculation:

    solFactor = ((Sol2VldDly + 6) * (FDSI / Fpixel)) + 6

    sol_delay =

    Ceil (TotalHorzPixelWidth * (FDSI / Fpixel)) -
    Ceil (((SplitFactor * TotalHorzPixelWidth) * BPP) /
    Gangedmode_lanes) + SolFactor;

    Final_sol_delay = (dcs_mode) ? (sol_delay + 20) : sol_delay

    Notes:

    - Sol2VldDly = Hsync start to Pixel valid delay
    - totalNoOfPixelsPerHorzLine = Total Number of horizontal pixel count.
    - SplitFactor = DSI_partition_lanes / Gangedmode_lanes.
    - Gangedmode_lanes = Total number of lanes required in Ganged mode transfer
    - DSI_partition_lanes = Total lanes in current partition/channel.
    - BPP = Bytes per pixel

Recommended Packet Sequence

Packet	Payload Size (Bytes)
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 8

- B = 2, 2.25 or 3, depending on pixel format.
- N = Number of Lanes used.

Table 91: Recommended Payload Size Table - Ganged Mode

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0								
1	HS	0								
2	HS	0								
3	HS	0	RGB	3	BLNK	4				
4	HS	0								
5	HS	0	RGB	3	BLNK	4				

Alternate Packet Sequence

Packet	Payload Size (Bytes)
ACTIVE	(T_{ACTIVE} * B)
HFP	(T_{HFP} * B) – 8

- B = 2, 2.25 or 3, depending on pixel format.
- N = Number of Lanes used.

Table 92: Optional Payload Size Table - Ganged Mode

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0								
1	HS	0								
2	HS	0								
3	HS	0	BLNK	2	RGB	3	BLNK	4		
4	HS	0								
5	HS	0	BLNK	2	RGB	3	BLNK	4		

26.4.15 Soft Reset Programming

Soft reset is asserted in the DSI through the LEG_DSI_ENABLE control field of the DSI_DSI_POWER_CONTROL_0 register. When soft reset is asserted, the DSI controller transitions the internal states to default mode. When it is asserted between

valid transactions, the current transaction is halted, the internal states are moved to default mode, and a new transaction starts by programming the desired values following the programming sequence explained in the other sections.

Note:	For Host transmissions enabled through the DSI_HOST_TRIGGER field of the DSI_DSI_TRIGGER_0 register, soft reset cannot be issued between a transaction, and software will write 1'b0 to the register to clear the trigger bit.

26.4.16 Function Programming

26.4.16.1 ECC Generation

For precise details on the calculation of the ECC field of the packet header, reference should be made to the MIPI Alliance Standard for Display Serial Interface. This subsection presents an overview of how the ECC can be created quite simply.

Each bit of the ECC byte is the result of XORing a number of bits from the packet header together. Which header bits contribute to which ECC bit is contained in a special table which can be used to calculate the ECC as follows:

```
const UCHAR ecc parity[24] = { 0x07, 0x0b, 0x0d, 0x0e, 0x13, 0x15, 0x16, 0x19,
                               0x1a, 0x1c, 0x23, 0x25, 0x26, 0x29, 0x2a, 0x2c,
                               0x31, 0x32, 0x34, 0x38, 0x1f, 0x2f, 0x37, 0x3b
                             };
ULONG packet header;
UCHAR ecc byte;
UINT  i;

// Assume bottom 24 bits of packet header
// contains header ID and byte count, then ...

ecc_byte = 0;
for (i = 0; i < 24; i++) {
  ecc_byte ^= ((packet_header >> i) & 1) ? ecc_parity[i] : 0x00;
}
packet_header |= (ULONG)(ecc_byte) << 24;
```

Note that the table in the DSI specification contains 64 entries. Since short packets have been fixed in length to be the same as a long packet header since the original specification was written, there will now always be just 24 data bits in the packet header, so only 24 entries are needed.

26.4.16.2 CRC Insertion

The DSI checksum used for long packets is derived using the generator polynomial $x^{16}+x^{12}+x^5+x^0$. Details of this can be found in section 9.6 of the MIPI DSI specification. To understand how the CRC is generated, think of the packet data as consisting of a continuous serialized stream of bits, rather than a sequence of 8-bit bytes. When thought of in this way, it is relatively straight forward to generate the CRC. Reproduced below is an abridged version of the example C code contained in Appendix B of the MIPI DSI specification.

```
// Polynomial, bit reversed form (since DSI transmits LSB first) …
const unsigned short CRC16GenerationCode = 0x8408;

unsigned short CalculateCRC16( unsigned char *DataStream ptr, unsigned short NumberOfDataBytes)

{
  unsigned short ByteCounter;
```

```
  unsigned char  BitCounter;
  unsigned char  CurrentData;
  unsigned short CRC16Result = 0xFFFF;

  if (NumberOfDataBytes > 0) {
    for (ByteCounter = 0; ByteCounter < NumberOfDataBytes; ByteCounteer++) {
      CurrentData = DataStream ptr[ByteCounter];
      for (BitCounter = 0; BitCounter < 8; BitCounter++) {
        if ((CRC16Result & 0x0001) ^ (CurrentData ^ 0x0001))
          CRC16Result = ((CRC16Result >> 1) & 0x7FFF) ^ CRCGenerationCode;
        else
          CRC16Result =  (CRC16Result >> 1) & 0x7FFF;
        CurrentData = (CurrentData >> 1) & 0x7F;
      }
    }
  }
  return CRC16Result;
}
```

This code may not be very high-performance due to the inner for loop which iterates over bits, rather than bytes. While it is instructive and could form the basis of a reference piece of code, it is not recommended where performance is important. There are many documented methods of performing this calculation in parallel in order to speed up the computations. These methods are beyond the scope of this document.

26.4.17 Read Data Return

DSI is a bidirectional interface. Data is returned from the peripheral display device only after the display controller has requested information by issuing a Bus Turn Around (BTA). All returned data is written into a FIFO that can be read using Host reads of a DSI register.

26.4.17.1 Reading Peripheral Registers

A typical application of a BTA is in the reading of a register from the display peripheral. This is achieved in the following way:

1. Set up the DSI interface to be in Host-driven command mode.
2. Set the DSI_MAX_THRESHOLD to 3.
3. Set the HOST_TX_TRIG_SRC field of the HOST_DSI_CONTROL register to FIFO_LEVEL.
4. Set the PKT_BTA field of the HOST_DSI_CONTROL register to ENABLE.
5. Write a DCS READ command packet (see section 8.8.8.2 of the MIPI DSI specification) into the Command FIFO by writing to the DSI_WR_DATA register.

This will result in the transmission of a DCS READ packet to the peripheral, the initiation of a BTA and – assuming the peripheral received the packet without error – the return of the requested data to the Host Read Return FIFO. The data can then be read from the FIFO by reading the DSI_RD_DATA register.

26.4.17.2 Bus Turn Around

- BTA is only supported for Host driven Command Mode interface. There will be no BTA during video mode transmission.
- Whether or not a BTA is initiated is controlled by the `PKT_BTA` field of the HOST_DSI_CONTROL register.
- The DSI Read Return FIFO is 4 bytes wide and 8 entries deep, so 32 bytes in total. This is enough to hold 8 short packets, or a mixture of short and long packets. The length of long packets must be severely restricted.

- Hardware does not perform ECC or CS checks on the read return data. Entire packets are simply made available to software to perform whatever checks they desire.
- There is the ability to increment a sync point counter on the arrival of the returned data or on the receipt of an error report in the event there was a problem with the read packet transaction.
- Software will ensure to program the byte clock frequency dsi_clk greater than 52 MHz, when performing BTA transactions.

26.4.17.3 BTA – Response Time Parameter

When a peripheral receives a READ Request, it is expected that a Bus Turn Around will immediately follow to extend support for the display device that cannot handle the request immediately after the Read request is issued.

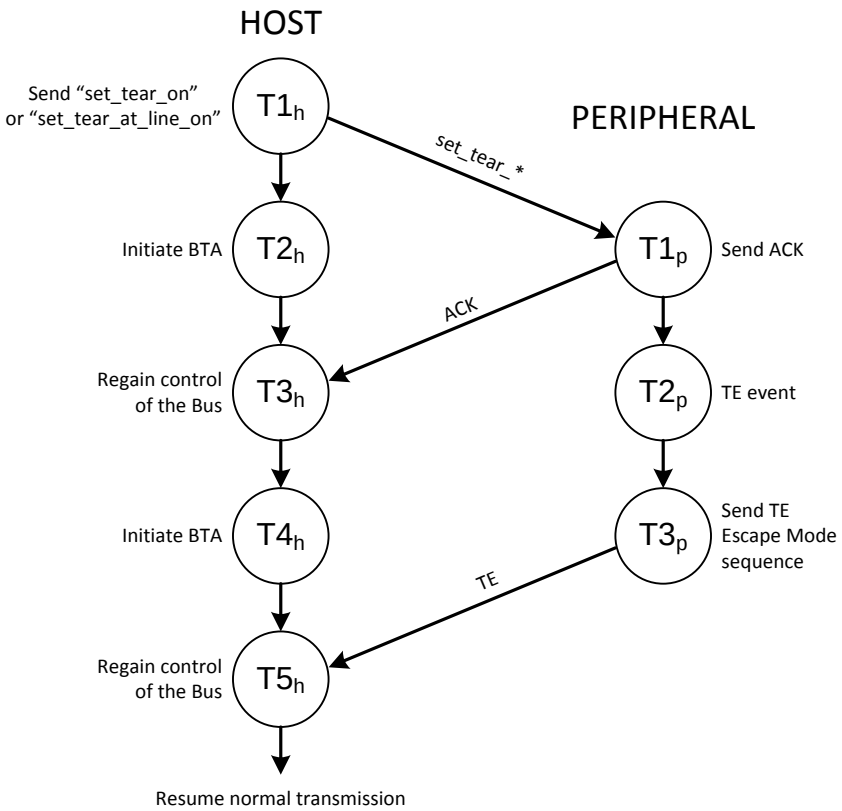
The DSI_DSI_BTA_TIMING_0 register defines DSI_TPKTBTA to allow a programmable delay between the end of host transmission and generation of BTA request for extending support for the display devices with slow response.

Note:	Valid programmable values for all the DSI_DSI_BTA_TIMING_0 register fields (DSI_TTAGO/DSI_TTASURE/DSI_TTAGET/DSI_TPKTBTA) are 0 to 254. Invalid value is 255.

26.4.18 Tearing Effect

In order to synchronize the update of a Command Mode display with data from the Host, a signal from the display can be sent to indicate when it is safe to proceed with the transmission of new data. This is the Tearing Effect reporting signal. Refer to section 8.12, “TE Signaling in DSI” of the DSI specification for more details.

Figure 117: Tearing Effect State Transition Diagram



Programmatically, this is achieved in the following way:

1. Send SET_TEAR_ON or SET_TEAR_AT_LINE_ON command with the PKT_BTA field in the HOST_DSI_CONTROL register set to ENABLE. This is state T1h.
2. Wait for ACK to come back from the peripheral. This is states T2h and T3h.
3. Set the IMM_BTA field in the HOST_DSI_CONTROL register set to ENABLE. This will cause the D-PHY to go into BTA without having to send a command first. This is state T4h.
4. Wait for TE return byte from the peripheral. This is state T5h.

Note: a trigger from panel/device to the Display Controller issued when a panel is ready to receive data from the Display Controller. The tear effect signal triggers when MSF/SSF register field in the “DC_CMD_DISPLAY_COMMAND_OPTION0_0” register is set to ENABLE. The Display Controller holds off a frame requested by input triggers until it receives a "ready" from the GPIO pins that connect to the tearing effect signals from the panel. It then sends a new frame to panel. In case no in-out trigger has been requested, the Display Controller shall not send a new frame.

26.4.19 Pad Calibration Programming

- Configure the mipi calibration clock to operate at 72 MHz via programming the CLK_RST_CONTROLLER_PLLP_BASE_0 register.
- Enable the MIPI calibration clock via Set CLK_ENB_MIPI_CAL to 1.
- Release reset to mipi_cal logic via deasserting SWR_MIPI_CAL_RST to 0
- Configure the DSI pad/bias pads to the appropriate values
  - DSI_PAD_CONTROL_3_0= 0x0
  - DSI_PAD_CONTROL_4_0 = 0x0
  - MIPI_CAL_MIPI_BIAS_PAD_CFG0_0 = 0x0
  - MIPI_CAL_MIPI_BIAS_PAD_CFG1_0 = 0x00020000
  - MIPI_CAL_MIPI_BIAS_PAD_CFG2_0 = 0x00000000
- Configure MIPI calibration settings for DSI pads for setup MIPI_CAL_DSI* _MIPI_CAL_CONFIG [*SELDSI*/ *OVERIDEDSI*/ *HSPDOSDSI*/ *HSPUOSDSI*/ *TERMOSDSI*]. MIPI_CAL_DSI*_MIPI_CAL_CONFIG_0
- Enable all the DSI lanes that require calibration driving LP_11 state.
- MIPI calibration start via - MIPI_CAL_MIPI_CAL_CTRL_0[MIPI_CAL_STARTCAL]
- Monitor the status via MIPI_CAL_CIL_MIPI_CAL_STATUS_0[*].

26.4.20 CRC Computation (Debug)

A CRC is computed on the byte data transmitted over the DSI lanes for debug purposes. Follow these steps for CRC generation:

1. Enable the DSI_DBG_ENABLE field in the DSI_DSI_CONTROL_0 register to turn on CRC computation.
2. When Video mode is enabled, CRC is generated from all bytes transmitted during a frame of video. Hardware will capture the CRC into a DSI_DSI_TX_CRC_0 register so that it can continue to calculate the CRC for the next frame without having to wait for software to read the current result.
3. In Host command mode, the CRC is generated from all bytes transmitted during the transaction. Software will latch the final CRC from the DSI_DSI_TX_CRC_0 register at the end of current host data transaction.

26.4.21 Error Reporting

There is no actual error recovery circuitry in the hardware, only error reporting. Any error that is reported via a protocol-level packet will be processed like all other return data and will be written to the data return FIFO for processing by the Host / Software.

Low level hardware errors such as bus contention will not be flagged, but will increment counters that can be queried by software so as to determine the reliability of the link.

26.4.21.1 Acknowledge with Error Report

The peripheral device (display) can be instructed to return an error report along with an acknowledge at the end of a transmission sequence. This is achieved by setting the PKT_BTA field in the HOST_DSI_CONTROL register to ENABLE. When this is done, there are several outcomes as shown in Table 93.

Table 93: Peripheral Response to Various Conditions

Condition	Non-Read Packet	Read Packet
No error	ACK	Read Data
Corrected single bit error	ACK with ERR	Read Data + ACK with ERR
Non-corrected multi-bit error	ACK with ERR	ACK with ERR
SOT, EOT, VI ID or other D-PHY error	ACK with ERR	ACK with ERR

ACK Return

If a BTA request is enabled and there is no error, then the peripheral will return an ACK trigger to the DSI hardware. This single byte trigger has the value 0x84. Since the return FIFO is 32 bits wide, the 0x84 value will be placed in the bottom 8 bits of the FIFO.

Read Data

If a read command packet is sent, then BTA request should be enabled to allow the peripheral to return the read data. If no error occurs in the transmission of the read packet, or a corrected single bit error occurs, then the read data will be returned to the host DSI hardware in the form of a read packet. The precise form will depend on the type of read packet transmitted. Refer to the MIPI DSI specification, section 8.10 for details.

If a corrected error occurs during the transmission of the read command, the requested read data will be returned in the normal way, but an ACK with Error report packet will be appended to the read return data packet.

ACK with Error Report

The error report is contained in the 16 payload bits of a special short packet returned by the peripheral. The bits allocations of the packet data are detailed in section 8.9.5 of the MIPI DSI specification but are repeated here for convenience.

Table 94: Error Report Bit Assignments

Bit	Description
0	SOT error
1	SOT sync error
2	EOT sync error
3	Escape Mode Entry Command error
4	Low-Power Transmit Sync Error
5	HS Receive Timeout error
6	False Control Error

Bit	Description
7	RESERVED
8	ECC Error, single bit (corrected)
9	ECC Error, multi-bit (not corrected)
10	CS Error (long packet only)
11	DSI data type not recognized
12	DSI Virtual Channel ID invalid
13	RESERVED
14	RESERVED
15	RESERVED

26.4.22 Time Outs

There are three compulsory and one optional time out counters required by the MIPI DSI specification for Processors (display controller). Each timer will consist of a simple counter which will count DSI byte clocks. The counters will reset to 0 on an event and will then simply increment their count every DSI byte clock cycle. If the event that the time out is protecting occurs before the time out reaches its terminal count, the counter will simply stop counting and hold its value. If the counter reaches software programmed maximum count before the expected event occurs, the counter will stop counting and hold its count value. Any action that is required by the MIPI DSI specification upon reaching the time out will also be performed by the hardware.

Table 95: Time-out Counter Summary

Time Out Name	Abbreviation	Start Condition	Length Greater Than	Action
HS Transmit TO	HTX_TO	SOT	Longest HS sequence	EOT + LP-11
LP Receive TO	LRXH_TO	LP Rx start	LTXP_TO (on periph.)	LP-11
Turn Around TO	TA_TO	BTA start	BTA response time	LP-11
Peripheral Reset	PR_TO	Reset Entry CMD	Peripheral resp. time	None.

In the table, the Peripheral Reset time out is the only timer that is optional. All the other timers are required by the MIPI DSI specification. Note that under “Action”, the listed operations are forced on the interface by the D-PHY under the instruction of the protocol layer (hardware). In the case of the optional Peripheral Reset time out, there is no action since this time out simply exists to issue a reset to the peripheral. Once the reset command is issued, the only further action required is to wait for an appropriate length of time to allow the peripheral to go through its reset sequence.

In the case of the HTX_TO, LRXH_TO and TA_TO time outs, the reaching of a terminal count will not only cause the action as listed in Table 95, but will also cause a tally counter to increment so that software can read the register and determine if any of the time outs have occurred. Software will be able to reset these tally counters to 0 by writing to the tally register. The value written will be irrelevant. The PR_TO will report its operation in a different manner. When the PR_TO counter is actually counting, there will be a status bit that reads as ‘0’. When the PR_TO terminal count is reached and time out ends, the status bit will become ‘1’. This will allow software to determine when the peripheral has been reset.

Table 96: Time Out and Tally Registers

Register	Field	Description
DSI_TIMEOUT_0	HTX_TO	High Speed Transmit time out duration
	LRXH_TO	Low Power Receive time out duration
DSI_TIMEOUT_1	TA_TO	Turn Around time out duration
	PR_TO	Peripheral Reset duration

Register	Field	Description
DSI_TO_TALLY	HTX_TALLY	High Speed Transmit time out Tally
	LRXH_TALLY	Low Power Receive time out Tally
	TA_TALLY	Turn Around time out Tally
	P_RESET_STATUS	Peripheral Reset Status: 0= Reset, 1 = Ready

26.5 MIPI-DSI Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

26.5.1 DSI_INCR_SYNCPT_0

Offset: 0x0 │ Byte Offset: 0x0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE 1 = OP_DONE 2 = RD_DONE 3 = REG_WR_SAFE 4 = COND_4 5 = COND_5 6 = COND_6 7 = COND_7 8 = COND_8 9 = COND_9 10 = COND_10 11 = COND_11 12 = COND_12 13 = COND_13 14 = COND_14 15 = COND_15
7:0	0x0	INDX: syncpt index value

26.5.2 DSI_INCR_SYNCPT_CNTRL_0

Offset: 0x1 │ Byte Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxx0)

Bit	Reset	Description
8	0x0	INCR_SYNCPT_NO_STALL: If NO_STALL is 1, then when FIFOs are full, INCR_SYNCPT methods will be dropped and the INCR_SYNCPT_ERROR[COND] bit will be set. If NO_STALL is 0, then when FIFOs are full, the client host interface will be stalled.
0	0x0	INCR_SYNCPT_SOFT_RESET: If SOFT_RESET is set, then all internal states of the client syncpt block will be reset. To do a soft reset, first set SOFT_RESET of all affected Host1x clients, then clear all SOFT_RESETs.

26.5.3 DSI_INCR_SYNCPT_ERROR_0

Offset: 0x2 │ Byte Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows. This bit is sticky and will remain set until cleared. Cleared by writing 1.

26.5.4 DSI_CTXSW_0

Context Switch Register

Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module, and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts.

Another way to avoid context switch interrupts is to set the AUTO_ACK bit. This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x8 │ Byte Offset: 0x20 │Read/Write: R/W │ Reset: 0xf000f800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

Note: **PACKET DEFINITIONS**

Packet for Display Controller -> DSI communication:

Line Type is used to convey the type of video line from the display controller to the DSI block. The line type implies the generation of one of the associated packet sequences as defined in the DSI packet sequence registers (see below). Used to construct packet headers. All packet headers are now 32-bits wide regardless of whether they are short or long packets. Used for validation infrastructure only.

26.5.5 DSI_DSI_RD_DATA_0

DSI Read Return Data

Offset: 0x9 | Byte Offset: 0x24 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_RD_DATA: Each read to this register will pop 32 bits from the 32-bit wide read return data FIFO. The FIFO has NV_DSI_HOST_DATA_RETURN_FIFO_DEPTH entries.

26.5.6 DSI_DSI_WR_DATA_0

Host FIFO Write Input

Offset: 0xa | Byte Offset: 0x28 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_WR_DATA: Each write to this register will push 32 bits into the 32-bit wide Host data FIFO. FIFO has NV_DSI_HOST_DATA_FIFO_DEPTH entries

26.5.7 DSI_DSI_POWER_CONTROL_0

Display Power Control

DSI Enable

Offset: 0xb | Byte Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	LEG_DSI_ENABLE: DSI interface Enable 0 = DISABLE : Disable DSI 1 = ENABLE : Enable DSI

26.5.8 DSI_INT_ENABLE_0

Interrupt Enable Register

Offset: 0xc | Byte Offset: 0x30 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	CTXSW_INT_ENABLE: Context Switch Interrupt Enable 0 = DISABLE : interrupt disabled 1 = ENABLE : interrupt enabled

26.5.9 DSI_INT_STATUS_0

Interrupt Status Register

Offset: 0xd | Byte Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CTXSW_INT: Context switch interrupt status (clear on write)

26.5.10 DSI_INT_MASK_0

Interrupt Mask

Setting bits in this register mask the corresponding interrupt but does not clear a pending interrupt and does not prevent a pending interrupt to be generated. Masking an interrupt also does not clear a pending interrupt status and does not prevent a pending interrupt status to be generated.

Offset: 0xe │ Byte Offset: 0x38 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CTXSW_INT_MASK: Context Switch Interrupt Mask 0 = MASKED : interrupt masked 1 = NOTMASKED : interrupt not masked

26.5.11 DSI_HOST_DSI_CONTROL_0

DSI Control Register When Input is from HOST

Offset: 0xf │ Byte Offset: 0x3c │ Read/Write: R/W │ Reset: 0x00000043 (0bxxxxxxxxxx00x000xx00xx0001000011)

Bit	Reset	Description
21	0x0	FIFO_STAT_RESET: write only bit to clear FIFO underflow/overflow flags. If a new underflow/overflow event occurs during the same time, current access cannot clear the status bits
20	0x0	CRC_RESET: Write only bit. When written with a 1, causes the Verification CRC generator to reset to 0xFFFF_FFFF. If written with 0, it has no effect.
18:16	0x0	DSI_PHY_CLK_DIV: Phy clock divider value for byte clock 0 = DIV1 1 = DIV2
13:12	0x0	HOST_TX_TRIG_SRC: Controls the source of the trigger to start sending packets. 0 = SOL : Start of Line signal from the Display Controller. 1 = FIFO_LEVEL : How full the FIFO is. Level determined elsewhere. 2 = IMMEDIATE : Determined by a write to the DSI_HOST_TRIGGER field of the DSI_TRIGGER register
9:8	0x0	DSI_ULTRA_LOW_POWER: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
7	0x0	PERIPH_RESET: Initiate an Escape Mode Peripheral Reset. 0 = DISABLE : Causes an Escape Mode Command to be sent to reset the 1 = ENABLE : External display device. Also starts the PR_TO counter. PR_TO state can be checked with the P_RESET_STATUS field in the DSI_TO_TALLY register. Hardware clears this bit upon completion of issuing "Trigger Reset" OR called "Reset Entry" command is sent out.
6	0x1	RAW_DATA: Host raw data mode. In this mode. All data is sent exactly as written. No attempt to decode packet headers is made. This bit will also override the function of the ECC and CS ENABLE fields. No ECC or CS will be generated. This mode is intended as a debug / diagnostic workaround mode only. 0 = DISABLE : Normal mode. 1 = ENABLE : Enable raw data transmission.
5	0x0	DSI_HIGH_SPEED_TRANS: DSI high speed transmission of packets 0 = LOW : Low speed - Note: Unlikely ever to be used. 1 = HIGH : High speed

Bit	Reset	Description
4	0x0	PKT_WR_FIFO_SEL: Host Write FIFO Select. In video mode, software shall not program PKT_WR_FIFO_SEL=VIDEO. 0 = HOST : Write data to the small host data FIFO only. 1 = VIDEO : Write data to both the host and video line store FIFO, in series. Note: Software shall only program PKT_WR_FIFO_SEL field in Host mode.
3	0x0	IMM_BTA: Generate BTA immediately, e.g., for Tearing Effect reporting. Note: This will generate a BTA and pass control of the D-PHY to the remote peripheral without the need to send any packet. Once the BTA is initiated on interface this bit gets cleared. Later on Host syncpt opdone is returned when the remote peripheral has responded and relinquished control of the bus. 0 = DISABLE : Do not generate BTA 1 = ENABLE : Generate BTA immediately, without waiting for a packet.
2	0x0	PKT_BTA: Generate BTA at the end of Host packets 0 = DISABLE : Do not generate BTA 1 = ENABLE : Generate BTA after the next packet is sent.
1	0x1	CS_ENABLE: enable Hardware Check Sum (CS) for Host packets Note: when CS is disabled, Host is responsible for generating proper CRC and adding the 2-byte CRC to the end of the packet after the payload. 0 = DISABLE : Disable hardware generation of CS (Host must calculate CS). 1 = ENABLE : Enable hardware generation of CS.
0	0x1	ECC_ENABLE: Enable hardware Error Correction Code (ECC) for Host packets Note: when ECC is disabled, Host is responsible for generating proper ECC byte for header 0 = DISABLE : Disable hardware generation of ECC (Host must calculate ECC). 1 = ENABLE : Enable hardware generation of ECC.

26.5.12 DSI_DSI_CONTROL_0

General DSI Control Register

Offset: 0x10 │ Byte Offset: 0x40 │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxxx0xx00xx00xx00xx000000)

Bit	Reset	Description
31	0x0	DSI_DBG_ENABLE: Control signal to turn off clock monitoring when enabled for debug, on every DSI byte clock debug signal toggle. Also TX CRC computation aid will turn on.
30	0x0	DFMT_16BPP_SWAP_EN: DSI specification supports different bit ordering (only 16 BPP) in command mode. Command Mode (Default) : 16BPP[15:0] -> B[4-0]G[5-0]R[4-0]; Command Mode (SWAP_EN) : 16BPP[15:0] -> G[2-0]B[4-0]R[4-0]G[5-3]
20	0x0	DSI_HS_CLK_CTRL: Control for the HS clock lane 0 = CONTINUOUS : HS clock is on all the time. 1 = TX_ONLY : HS clock is only active during HS transmissions.
17:16	0x0	DSI_VIRTUAL_CHANNEL: Virtual channel ID. The virtual channel is sent as part of the packet header and used to distinguish multiple displays.
13:12	0x0	DSI_DATA_FORMAT: Pixel Data format transmitted. Note that although the pixel format is specified in the packet header ID for RGB data packets, this information is ignored by the hardware. Only the information used in this register is used in the construction of RGB data packets. 0 = BIT16P : 16 bpp RGB Packed. 2 bytes used per pixel 1 = BIT18NP : 18 bpp RGB Not-packed. 3 bytes used per pixel 2 = BIT18P : 18 bpp RGB Packed. 2.25 bytes used per pixel 3 = BIT24P : 24 bpp RGB Packed. 3 bytes used per pixel
9:8	0x0	VID_TX_TRIG_SRC: Controls the source of the trigger to start sending packets. 0 = SOL : Start of Line signal from the Display Controller. 1 = FIFO_LEVEL : How full the FIFO is. Level determined elsewhere. 2 = IMMEDIATE : Determined by a write to the DSI_VID_TRIGGER field of the DSI_TRIGGER register

Bit	Reset	Description
5:4	0x0	DSI_NUM_DATA_LANES: Number of D-PHY data lanes used by Display for HS transmission. 0 = ONE : 1 data lane. 1 = TWO : 2 data lanes. 2 = THREE : 3 data lanes. 3 = FOUR : 4 data lanes.
3	0x0	VID_DCS_ENABLE: Enable for insertion of DCS commands during Display Controller generated packets. When enabled, the DCS commands defined in the LT3_DCS_CMD and LT5_DCS_CMD fields of the DSI_DCS_CMDS register will be inserted in long packets defined in packet sequence 3 and 5. 0 = DISABLE : No DCS commands will be inserted. 1 = ENABLE : DCS command IDs will be inserted as described above.
2	0x0	DSI_VID_SOURCE: Source of video pixels 0 = DISPLAY_0 : Pixels come from "display" 1 = DISPLAY_1 : Pixels come from "displayb"
1	0x0	DSI_VID_ENABLE: Video DSI Interface Enable 0 = DISABLE : Disable 1 = ENABLE : Enable
0	0x0	DSI_HOST_ENABLE: Host DSI Interface Enable 0 = DISABLE : Disable 1 = ENABLE : Enable

26.5.13 DSI_DSI_SOL_DELAY_0

Number of Byte-Clock Counts to Wait after Reception

Offset: 0x11 | Byte Offset: 0x44 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SOL_DELAY: Start Of Line before generating output packets.

26.5.14 DSI_DSI_MAX_THRESHOLD_0

Maximum Threshold Registers for DSI Related Packets

Offset: 0x12 | Byte Offset: 0x48 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	MAX_THRESHOLD: Start draining FIFO once this threshold is met. This register can be used for DBI mode when line packet data exceeds the size of the data FIFO.

26.5.15 DSI_DSI_TRIGGER_0

Manual Transmission Trigger Register

Offset: 0x13 | Byte Offset: 0x4c | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
1	X	DSI_HOST_TRIGGER: A 1 written to this bit will start host transmission when HOST_DSI_CONTROL.HOST_TX_TRIG_SRC is set to IMMEDIATE. Hardware auto clears on operation completion
0	X	DSI_VID_TRIGGER: A 1 written to this bit will start video transmission when DSI_CONTROL.VID_TX_TRIG_SRC is set to IMMEDIATE. Hardware auto clears on operation completion

26.5.16 DSI_DSI_TX_CRC_0

Transmission CRC

Offset: 0x14 | Byte Offset: 0x50 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	TX_CRC: Long Packet CRC appended to the end of long packets. This CRC is that result of generating a CRC from all transmitted bytes. If DSI_HOST_ENABLE == ENABLE, then a CRC is generated from all bytes transmitted from the Host interface in Command Mode. If DSI_VID_ENABLE == ENABLE, then a CRC is generated from all bytes transmitted during a frame of video when in Video Mode. Note that the hardware will capture the CRC into a separate internal register so that it can continue to calculate the CRC for the next frame without having to wait for software to read the current result.

26.5.17 DSI_DSI_STATUS_0

DSI Status Register

Offset: 0x15 | Byte Offset: 0x54 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
10	X	DSI_IDLE: Indicated that the DSI is IDLE.
9	X	LB_UNDERFLOW: Indicates that a Line buffer underflow event happened
8	X	LB_OVERFLOW: Indicates that a Line buffer overflow event happened
4:0	X	RD_FIFO_COUNT: Count of how many data words are left in the Host Read Data Return FIFO. Typically, software knows how much data to read from the DSI_RD_DATA register after a Read packet / BTA has been sent / requested, since these transactions are initiated by software. However, under error conditions, insufficient data may have been read from the FIFO. Software should therefore check this field to make sure it is 0 after reading all the information it expected to get.

26.6 Initialization Sequence Registers

26.6.1 DSI_DSI_INIT_SEQ_CONTROL_0

DSI Initialization Sequence Control

Offset: 0x1a | Byte Offset: 0x68 | Read/Write: R/W | Reset: 0x0000XX00 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
14:8	X	DSI_FRAME_INIT_BYTE_COUNT: Frame Initialization Sequence Byte Count. This parameter specifies the number of frame initialization sequence cycles to send. If programmed to 0, there is no frame initialization cycle generated. Valid programmable values: 0 to 64. Invalid programmable values: 65 to 127.
0	0x0	DSI_SEND_INIT_SEQUENCE: Send Initialization Sequence (IS) 0 = DISABLE 1 = ENABLE

26.6.2 DSI_DSI_INIT_SEQ_DATA_0_0

DSI Init Sequence Write Data 0

Offset: 0x1b | Byte Offset: 0x6c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_0: DSI Init Sequence Write Data bits 31:0

26.6.3 DSI_DSI_INIT_SEQ_DATA_1_0

DSI Init Sequence Write Data 1

Offset: 0x1c | Byte Offset: 0x70 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_1: DSI Init Sequence Write Data bits 31:0

26.6.4 DSI_DSI_INIT_SEQ_DATA_2_0

DSI Init Sequence Write Data 2

Offset: 0x1d | Byte Offset: 0x74 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_2: DSI Init Sequence Write Data bits 31:0

26.6.5 DSI_DSI_INIT_SEQ_DATA_3_0

DSI Init Sequence Write Data 3

Offset: 0x1e | Byte Offset: 0x78 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_3: DSI Init Sequence Write Data bits 31:0

26.6.6 DSI_DSI_INIT_SEQ_DATA_4_0

DSI Init Sequence Write Data 4

Offset: 0x1f | Byte Offset: 0x7c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_4: DSI Init Sequence Write Data bits 31:0

26.6.7 DSI_DSI_INIT_SEQ_DATA_5_0

DSI Init Sequence Write Data 5

Offset: 0x20 | Byte Offset: 0x80 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_5: DSI Init Sequence Write Data bits 31:0

26.6.8 DSI_DSI_INIT_SEQ_DATA_6_0

DSI Init Sequence Write Data 6

Offset: 0x21 | Byte Offset: 0x84 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_6: DSI Init Sequence Write Data bits 31-0

26.6.9 DSI_DSI_INIT_SEQ_DATA_7_0

DSI Init Sequence Write Data 7

Offset: 0x22 | Byte Offset: 0x88 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_7: DSI Init Sequence Write Data bits 31-0

26.7 Packet Sequence Registers

These registers allow the construction of arbitrary packet sequences associated with various video line types as sent from the Display Controller to the DSI block.

The first digit of the pair of digits in each field below is the sequence number, and the second digit denotes the packet number within each sequence. For example, field PKT_23_ID, defines the ID for packet 3 in sequence 2.

26.7.1 DSI_DSI_PKT_SEQ_0_LO_0

DSI Packet Sequence 0 LO Half

Offset: 0x23 | Byte Offset: 0x8c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_0_FORCE_LP: For packet sequence 0, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_02_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_02_ID: Packet 2 Packet ID
22:20	X	PKT_02_SIZE: Packet 2 size pointer
19	X	PKT_01_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_01_ID: Packet 1 Packet ID
12:10	X	PKT_01_SIZE: Packet 1 size pointer
9	X	PKT_00_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_00_ID: Packet 0 Packet ID
2:0	X	PKT_00_SIZE: Packet 0 size pointer

26.7.2 DSI_DSI_PKT_SEQ_0_HI_0

DSI Packet Sequence 0 HI Half

Line Type 0 is associated with the first line in the frame and should contain a packet sequence that starts with a VS packet.

Offset: 0x24 │ Byte Offset: 0x90 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_05_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_05_ID: Packet 5 Packet ID
22:20	X	PKT_05_SIZE: Packet 5 size pointer
19	X	PKT_04_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_04_ID: Packet 4 Packet ID
12:10	X	PKT_04_SIZE: Packet 4 size pointer
9	X	PKT_03_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_03_ID: Packet 3 Packet ID
2:0	X	PKT_03_SIZE: Packet 3 size pointer

26.7.3 DSI_DSI_PKT_SEQ_1_LO_0

DSI Packet Sequence 1 LO Half

Line Type 1 is associated with the last line of Vertical Sync and should contain a packet sequence that starts with a VE packet.

Offset: 0x25 │ Byte Offset: 0x94 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_1_FORCE_LP: For packet sequence 1, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_12_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_12_ID: Packet 2 Packet ID
22:20	X	PKT_12_SIZE: Packet 2 size pointer
19	X	PKT_11_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_11_ID: Packet 1 Packet ID
12:10	X	PKT_11_SIZE: Packet 1 size pointer
9	X	PKT_10_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_10_ID: Packet 0 Packet ID

Bit	Reset	Description
2:0	X	PKT_10_SIZE: Packet 0 size pointer

26.7.4 DSI_DSI_PKT_SEQ_1_HI_0

DSI Packet Sequence 1 HI Half

Offset: 0x26 | Byte Offset: 0x98 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_15_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_15_ID: Packet 5 Packet ID
22:20	X	PKT_15_SIZE: Packet 5 size pointer
19	X	PKT_14_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_14_ID: Packet 4 Packet ID
12:10	X	PKT_14_SIZE: Packet 4 size pointer
9	X	PKT_13_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_13_ID: Packet 3 Packet ID
2:0	X	PKT_13_SIZE: Packet 3 size pointer

26.7.5 DSI_DSI_PKT_SEQ_2_LO_0

DSI Packet Sequence 2 LO Half

Line Type 2 is associated with any vertical blank line except the first one after active and should contain a packet sequence that starts with an HS packet.

Offset: 0x27 | Byte Offset: 0x9c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_2_FORCE_LP: For packet sequence 2, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_22_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_22_ID: Packet 2 Packet ID
22:20	X	PKT_22_SIZE: Packet 2 size pointer
19	X	PKT_21_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_21_ID: Packet 1 Packet ID
12:10	X	PKT_21_SIZE: Packet 1 size pointer

Bit	Reset	Description
9	X	PKT_20_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_20_ID: Packet 0 Packet ID
2:0	X	PKT_20_SIZE: Packet 0 size pointer

26.7.6 DSI_DSI_PKT_SEQ_2_HI_0

DSI Packet Sequence 2 HI Half

Offset: 0x28 | Byte Offset: 0xa0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_25_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_25_ID: Packet 5 Packet ID
22:20	X	PKT_25_SIZE: Packet 5 size pointer
19	X	PKT_24_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_24_ID: Packet 4 Packet ID
12:10	X	PKT_24_SIZE: Packet 4 size pointer
9	X	PKT_23_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_23_ID: Packet 3 Packet ID
2:0	X	PKT_23_SIZE: Packet 3 size pointer

26.7.7 DSI_DSI_PKT_SEQ_3_LO_0

DSI Packet Sequence 3 LO Half

Line Type 3 is associated with any active line except the first one and should contain a packet sequence that starts with an HS packet and includes an RGB data packet.

Offset: 0x29 | Byte Offset: 0xa4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_3_FORCE_LP: For packet sequence 3, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_32_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_32_ID: Packet 2 Packet ID
22:20	X	PKT_32_SIZE: Packet 2 size pointer
19	X	PKT_31_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:13	X	PKT_31_ID: Packet 1 Packet ID
12:10	X	PKT_31_SIZE: Packet 1 size pointer
9	X	PKT_30_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_30_ID: Packet 0 Packet ID
2:0	X	PKT_30_SIZE: Packet 0 size pointer

26.7.8 DSI_DSI_PKT_SEQ_3_HI_0

DSI Packet Sequence 3 HI Half

Offset: 0x2a │ Byte Offset: 0xa8 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29	X	PKT_35_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_35_ID: Packet 5 Packet ID
22:20	X	PKT_35_SIZE: Packet 5 size pointer
19	X	PKT_34_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_34_ID: Packet 4 Packet ID
12:10	X	PKT_34_SIZE: Packet 4 size pointer
9	X	PKT_33_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_33_ID: Packet 3 Packet ID
2:0	X	PKT_33_SIZE: Packet 3 size pointer

26.7.9 DSI_DSI_PKT_SEQ_4_LO_0

DSI Packet Sequence 4 LO Half

Line Type 4 is associated with the first vertical blanking line after the last active line and should contain a packet sequence that starts with an HS packet. Ordinarily, this packet sequence should be identical to sequence 2.

Offset: 0x2b │ Byte Offset: 0xac │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
30	X	SEQ_4_FORCE_LP: For packet sequence 4, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_42_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_42_ID: Packet 2 Packet ID
22:20	X	PKT_42_SIZE: Packet 2 size pointer

Bit	Reset	Description
19	X	PKT_41_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_41_ID: Packet 1 Packet ID
12:10	X	PKT_41_SIZE: Packet 1 size pointer
9	X	PKT_40_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_40_ID: Packet 0 Packet ID
2:0	X	PKT_40_SIZE: Packet 0 size pointer

26.7.10 DSI_DSI_PKT_SEQ_4_HI_0

DSI Packet Sequence 4 HI Half

Offset: 0x2c │ Byte Offset: 0xb0 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_45_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_45_ID: Packet 5 Packet ID
22:20	X	PKT_45_SIZE: Packet 5 size pointer
19	X	PKT_44_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_44_ID: Packet 4 Packet ID
12:10	X	PKT_44_SIZE: Packet 4 size pointer
9	X	PKT_43_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_43_ID: Packet 3 Packet ID
2:0	X	PKT_43_SIZE: Packet 3 size pointer

26.7.11 DSI_DSI_PKT_SEQ_5_LO_0

DSI Packet Sequence 5 LO Half

Line Type 5 is associated with the first active line. It should contain a packet sequence that starts with an HS packet and includes an RGB data packet. Ordinarily, this packet sequence should be identical to sequence 3.

Offset: 0x2d │ Byte Offset: 0xb4 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
30	X	SEQ_5_FORCE_LP: For packet sequence 5, forces the D-PHY to go to LP mode after the last packet of the sequence has been transmitted. 0 = DISABLE 1 = ENABLE
29	X	PKT_52_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28:23	X	PKT_52_ID: Packet 2 Packet ID
22:20	X	PKT_52_SIZE: Packet 2 size pointer
19	X	PKT_51_EN: Packet 1 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_51_ID: Packet 1 Packet ID
12:10	X	PKT_51_SIZE: Packet 1 size pointer
9	X	PKT_50_EN: Packet 0 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_50_ID: Packet 0 Packet ID
2:0	X	PKT_50_SIZE: Packet 0 size pointer

26.7.12 DSI_DSI_PKT_SEQ_5_HI_0

DSI Packet Sequence 5 HI Half

Offset: 0x2e │ Byte Offset: 0xb8 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
29	X	PKT_55_EN: Packet 5 enable 0 = DISABLE 1 = ENABLE
28:23	X	PKT_55_ID: Packet 5 Packet ID
22:20	X	PKT_55_SIZE: Packet 5 size pointer
19	X	PKT_54_EN: Packet 4 enable 0 = DISABLE 1 = ENABLE
18:13	X	PKT_54_ID: Packet 4 Packet ID
12:10	X	PKT_54_SIZE: Packet 4 size pointer
9	X	PKT_53_EN: Packet 3 enable 0 = DISABLE 1 = ENABLE
8:3	X	PKT_53_ID: Packet 3 Packet ID
2:0	X	PKT_53_SIZE: Packet 3 size pointer

26.8 DCS Command and Packet Length Registers

26.8.1 DSI_DSI_DCS_CMDS_0

DCS command IDs used for Line Types 3 and 5. These command IDs are inserted at the start of the data payload of DCS long packets when the Display Controller is being used to transmit DCS commands.

Offset: 0x33 │ Byte Offset: 0xcc │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
15:8	X	LT5_DCS_CMD: DCS command for Line Type 5.
7:0	X	LT3_DCS_CMD: DCS command for Line Type 3.

26.8.2 DSI_DSI_PKT_LEN_0_1_0

DSI Packet Lengths 0 and 1

Offset: 0x34 | Byte Offset: 0xd0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LENGTH_1: Packet length 1 (in bytes)
15:0	X	LENGTH_0: Packet length 0 (in bytes)

26.8.3 DSI_DSI_PKT_LEN_2_3_0

DSI Packet Lengths 2 and 3

Offset: 0x35 | Byte Offset: 0xd4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LENGTH_3: Packet length 3 (in bytes)
15:0	X	LENGTH_2: Packet length 2 (in bytes)

26.8.4 DSI_DSI_PKT_LEN_4_5_0

DSI Packet Lengths 4 and 5

Offset: 0x36 | Byte Offset: 0xd8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LENGTH_5: Packet length 5 (in bytes)
15:0	X	LENGTH_4: Packet length 4 (in bytes)

26.8.5 DSI_DSI_PKT_LEN_6_7_0

DSI Packet Lengths 6 and 7

Offset: 0x37 | Byte Offset: 0xdc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LENGTH_7: Packet length 7 (in bytes)
15:0	X	LENGTH_6: Packet length 6 (in bytes)

26.9 Physical Interface Timing Registers

The DSI PHY timing registers must be initialized before the interface is enabled. Based on the equations provided in the table below, all PHY timing parameters should be programmed in the timing registers. T_{byte} refers to just the DSI controller clock.

Table 97 D-PHY Timing Values

Parameter	MIPI D-PHY Spec Value	Programmed Value
THS-PREPARE	(40 ns + 4*UI , 85 ns + 6*UI)	Min - (40 + (4 T_{bit}))) / T_{byte} Max - ((85 + 6 T_{bit})) / T_{byte}

Parameter	MIPI D-PHY Spec Value	Programmed Value
THS-ZERO	105 ns + 6*UI min.	(105 + (6 T_{bit})) / T_{byte} -2
THS-TRAIL	max(8*UI, 60 ns + 4*UI)	3 + max(8 T_{bit}, 60 + 4 T_{bit}) / T_{byte}
THS-EXIT	100 ns min.	100 / T_{byte}
TLPX	50 ns min.	50 / T_{byte}
TCLK-ZERO	300 ns along with TCLK-PREPARE	260/ T_{byte}
TCLK-PREPARE	(38 ns, 95 ns)	THS-PREPARE timing should guarantee to meet both these timing requirements
TCLK-POST	60 ns + 52*UI min.	(60 + (52 T_{bit})) / T_{byte} - 2
TCLK-TRAIL	60 ns min.	60 / T_{byte}
TCLK-PRE	8 UI	8 T_{bit}
TTA-GO	4 * TLPX	3 * TLPX
TTA-SURE	(TLPX, 2 * TLPX)	TLPX - 1
TTA-GET	5 * TLPX	4 * TLPX
TWAKEUP	1 ms	1 ms/10 ns x 512 {in terms of 512 byte clocks}

Notes:

- T_{byte} = Period of one byte clock in nanoseconds.
- T_{bit} = Period of one bit time in nanoseconds. (T_{bit} = T_{byte} / 8)
- All figures and time periods are in nanoseconds.
- All timing register values are multiples of byte clock period, T_{byte}.
- All fractional results are truncated. No rounding.

26.9.1 DSI_DSI_PHY_TIMING_0_0

DSI D-PHY Timing Register 0

Offset: 0x3c │ Byte Offset: 0xf0 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	DSI_THSDEXIT: Time to drive LP11 after HS **Note:** Valid programmable values for all the fields in this register (i.e., DSI_THSPREPR/DSI_TDATZERO/DSI_THSTRAIL/DSI_THSDEXIT) are 0 to 254. Invalid value is 255.
23:16	X	DSI_THSTRAIL: Time to drive HS flipped bit at EOT
15:8	X	DSI_TDATZERO: Time to drive HS0 before SOT
7:0	X	DSI_THSPREPR: Time to drive LP00 before HS data

26.9.2 DSI_DSI_PHY_TIMING_1_0

DSI D-PHY Timing Register 1

Offset: 0x3d │ Byte Offset: 0xf4 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	DSI_TCLKTRAIL: Time to drive HS0 before the clock goes to LP1 **Note:** Valid programmable values for all the fields in this register (i.e., DSI_TTLPX/DSI_TCLKZERO/DSI_TCLKPOST/ DSI_TCLKTRAIL) are 0 to 254 Invalid value is 255.
23:16	X	DSI_TCLKPOST: Time to drive clock after the last HS data
15:8	X	DSI_TCLKZERO: Time to drive LP00 before HS clock
7:0	X	DSI_TTLPX: LP period

26.9.3 DSI_DSI_PHY_TIMING_2_0

DSI D-PHY Timing Register 2

Offset: 0x3e │ Byte Offset: 0xf8 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
23:16	X	DSI_TCLKPREPARE: Time to drive LP0 before CLK_ZERO starts off on the clock lane. **Note:** Valid programmable values for the following register fields (i.e., DSI_TCLKPRE/DSI_TCLKPREPARE) are 0 to 254. Invalid value is 255
15:8	X	DSI_TCLKPRE: Time to run clock before enabling data lane
7:0	X	DSI_TWAKEUP: LP period when exiting ULPM, in units of 512 byte clocks.

26.9.4 DSI_DSI_BTA_TIMING_0

DSI D-PHY Bus-Turn-Around Timing

Offset: 0x3f │ Byte Offset: 0xfc │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	DSI_TPKTBTA: Programmable Time delay between end of Host packet transmission and generation of Pkt BTA in PKT_BTA mode (i.e., HOST_DSI_CONTROL[2]) - Generate BTA at the end of Host packets **Note**: Valid programmable values for all the fields in this register (i.e., DSI_TTAGO/DSI_TTASURE/DSI_TTAGET/ DSI_TPKTBTA) are 0 to 254. Invalid value is 255.
23:16	X	DSI_TTAGET: Time to Drive LP00 at end of BTA (5 * TTLPX)
15:8	X	DSI_TTASURE: Time to Receive LP00 at end of BTA (2 * TTLPX)
7:0	X	DSI_TTAGO: Time to drive LP00 at start of BTA (4 * TTLPX)

26.10 Contention Recovery Timers

These registers control the length of time - in units of 512 DSI byte clocks – that can elapse before the hardware will decide that a bus contention error has occurred.

There are several counters to deal with various forms of error that may occur. For details, refer to the MIPI DSI Specification, section 7.2.2.

26.10.1 DSI_DSI_TIMEOUT_0_0

DSI Time Out Terminal Count Register 0

Offset: 0x44 | Byte Offset: 0x110 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	LRXH_TO: Low Power Receive (Host) Time Out terminal count
15:0	X	HTX_TO: High Speed Transmit Time Out terminal count

26.10.2 DSI_DSI_TIMEOUT_1_0

DSI Time Out Terminal Count Register 1

Offset: 0x45 | Byte Offset: 0x114 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	PR_TO: Peripheral Reset duration.
15:0	X	TA_TO: Turn Around Time Out terminal count

26.10.3 DSI_DSI_TO_TALLY_0

DSI Time Out Tally Register

Each time one of the time-out counters reaches its terminal count, it will increment the associated tally register.

Offset: 0x46 | Byte Offset: 0x118 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	R/W	Reset	Description
24	RO	X	P_RESET_STATUS: Peripheral Reset time out status 0 = IN_RESET 1 = READY
23:16	RW	X	TA_TALLY: Turn Around time out tally
15:8	RW	X	LRXH_TALLY: LP Rx time out tally
7:0	RW	X	HTX_TALLY: HS Tx time out tally

26.11 Physical Pad Control Registers

26.11.1 DSI_PAD_CONTROL_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x4b │ Byte Offset: 0x12c │ Read/Write: R/W │ Reset: 0x010f010f (0bxxxxxxx1xxxx1111xxxxxxx1xxxx1111)

Bit	Reset	Description
24	0x1	DSI_PAD_PULLDN_CLK_ENAB:1= Enable pad pulldown for clock bit at power on
19:16	0xf	DSI_PAD_PULLDN_ENAB:1= Enable pad pulldown for data bits at power on
8	0x1	DSI_PAD_PDIO_CLK: Power down for clock bit, drivers, receivers, and contention detectors
3:0	0xf	DSI_PAD_PDIO: Power down for data bit, drivers, receivers, and contention detectors

26.11.2 DSI_PAD_CONTROL_CD_0

Contention Detection Logic Enable Signals

Offset: 0x4c │ Byte Offset: 0x130 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxx000xxxxxxx0xxxx0000)

Bit	Reset	Description
18:16	0x0	DSI_PAD_CDDNADJ: Level adjust on low limit of detection. Tie to 0 (DSI emu, miniTMC control). 000 = 0.3V 001 = 0.375V 010 = 0.45V 011 = 0.525V 100 = 0.3V 101 = 0.225V 110 = 0.15V 111 = 0.075V
8	0x0	DSI_PAD_CD_EN_CLK: Clock bit contention detector enable. 1 = Enable
3:0	0x0	DSI_PAD_CD_EN: Data bits contention detector enable. 1 = EnableCD_EN_[1] = 0 in DSI.

26.11.3 DSI_PAD_CD_STATUS_0

Contention Detection Status from MIPI PAD

Offset: 0x4d │ Byte Offset: 0x134 │ Read/Write: RO │ Reset: 0x000X0X0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
18	X	DSI_PAD_CDN_CLK
16	X	DSI_PAD_CDP_CLK
11:8	X	DSI_PAD_CDN
3:0	X	DSI_PAD_CDP

26.11.4 DSI_DSI_VID_MODE_CONTROL_0

Host Command Packet During Video Mode

Offset: 0x4e │ Byte Offset: 0x138 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:1	0x0	DSI_LINE_TYPE: Vertical blank. LINE TYPE on which the host command packet is to be transmitted, i.e., valid values are 0/1/2/4 0 = ZERO : Line type 0 1 = ONE : Line type 1 2 = TWO : Line type 2 3 = THREE : NA: Host command packets on line type 3 are invalid 4 = FOUR : Line type 4 5 = FIVE : NA: Host command packets on line type 5 are invalid 6 = SIX : NA: Host command packets on line type 6 are invalid 7 = SEVEN : NA: Host command packets on line type 7 are invalid
0	0x0	DSI_CMD_PKT_VID_ENABLE: 0 = DISABLE : Disable host command packet during video mode 1 = ENABLE : Enable host command packet during video mode

26.11.5 DSI_PAD_CONTROL_1_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x4f │ Byte Offset: 0x13c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx000x000x000x000)

Bit	Reset	Description
14:12	0x0	DSI_PAD_OUTADJ3: Input delay trimmer for data bit 3. Each tap delays 40 ps
10:8	0x0	DSI_PAD_OUTADJ2: Input delay trimmer for data bit 2. Each tap delays 40 ps
6:4	0x0	DSI_PAD_OUTADJ1: Input delay trimmer for data bit 1. Each tap delays 40 ps
2:0	0x0	DSI_PAD_OUTADJ0: Input delay trimmer for data bit 0.Each tap delays 40 ps

26.11.6 DSI_PAD_CONTROL_2_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x50 │ Byte Offset: 0x140 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxx000x000x000x000x000)

Bit	Reset	Description
18:16	0x0	DSI_PAD_SLEWUPADJ: Pull-up slew rate adjust. 000 = 011: slew rate increases 100 = 000, 100 =111: skew rate decreases
14:12	0x0	DSI_PAD_SLEWDNADJ: Pull-down slew rate adjust. 000 = 011, slew rate increases 100 = 000, 100 = 111: skew rate decreases.
10:8	0x0	DSI_PAD_LPUPADJ: Driver pull-up impedance control. 00 = 130 ohms, default, 01 = 110 ohms 10 = 130 ohms 11 = 150 ohms

Bit	Reset	Description
6:4	0x0	DSI_PAD_LPDNADJ: Driver pull-down impedance control. 00 = 130 ohms, default 01 = 110 ohms 10 = 130 ohms, same as 0 011 = 150 ohms
2:0	0x0	DSI_PAD_OUTADJCLK: Output trimmer delay for clock bit. Each tap delays 40 ps

26.11.7 DSI_PAD_CONTROL_3_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x51 │ Byte Offset: 0x144 │ Read/Write: R/W │ Reset: 0x10000000 (0bxxx1xxxxxxxxxxx0xx00xx00xx00xx00)

Bit	Reset	Description
28	0x1	DSI_PAD_PDVCLAMP: Power down regulator which supplies current to serializer/deserializer logic. Active high, 1=power down
16	0x0	DSI_PAD_BANDWD_IN: Increase bandwidth of differential receiver
13:12	0x0	DSI_PAD_PREEMP_PD_CLK: Clock bit HS driver pull down pre-emphasis. 00 = no_preemphasis 11= max
9:8	0x0	DSI_PAD_PREEMP_PU_CLK: Clock bit HS driver pull up pre-emphasis. 00 = no_preemphasis 11= max
5:4	0x0	DSI_PAD_PREEMP_PD: Data bit HS driver pull down pre-emphasis. 00 = no_preemphasis 11= max
1:0	0x0	DSI_PAD_PREEMP_PU: Data bit HS driver pull up pre-emphasis. 00 = no_preemphasis 11= max

26.11.8 DSI_PAD_CONTROL_4_0

DSI PHY Configuration Register

Software can program this register to work with different panels.

Offset: 0x52 │ Byte Offset: 0x148 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxx0xxxx0000xxx0xxxx0000xxx0xxx0)

Bit	Reset	Description
28	0x0	DSI_PAD_HS_BSO_CLK:1= Enables BIAS and power regulators on for HS mode
23:20	0x0	DSI_PAD_HS_BSO:1= Enables BIAS and power regulators on for HS mode
16	0x0	DSI_PAD_LP_BSO_CLK:1= Enables BIAS and power regulators on for LP mode
11:8	0x0	DSI_PAD_LP_BSO:1= Enables BIAS and power regulators on for LP mode
4	0x0	DSI_PAD_TXBW_EN:1= Increase bandwidth of output driver, default=0
0	0x0	DSI_PAD_REV_CLK:1= Reverse clock polarity

26.11.9 DSI_DSI_GANGED_MODE_CONTROL_0

Offset: 0x53 | Byte Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	DSI_GANGED_MODE_EN: 0 = DISABLE : Ganged mode transaction disabled 1 = ENABLE : Ganged mode transaction enabled

26.11.10 DSI_DSI_GANGED_MODE_START_0

Offset: 0x54 | Byte Offset: 0x150 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx0000000000000)

Bit	Reset	Description
12:0	0x0	DSI_GANGED_START_POINTER: Start pointer for indicating the start of partial active valid pixel data to be latched from the valid pixels of the display controller in Left-Right/Even-Odd ganged mode. Note: Program to the default value/zero when DSI_GANGED_MODE_EN = DISABLE (non ganged mode).

26.11.11 DSI_DSI_GANGED_MODE_SIZE_0

Offset: 0x55 | Byte Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0bxxx0000000000000xxx0000000000000)

Bit	Reset	Description
28:16	0x0	DSI_GANGED_VALID_LOW_WIDTH: Width of Partial Inactive/Ignored pixel data from the valid pixels of the display controller in Left-Right/Even-Odd ganged mode. Note: Program to the default value/zero when DSI_GANGED_MODE_EN = DISABLE (non-ganged mode).
12:0	0x0	DSI_GANGED_VALID_HIGH_WIDTH: Width of Partial Active valid pixel data latched from the valid pixels of display controller in Left-Right/Even-Odd ganged mode. Note: Program to the default value/zero when DSI_GANGED_MODE_EN = DISABLE (non-ganged mode).

26.11.12 DSI_DSI_RAW_DATA_BYTE_COUNT_0

Offset: 0x56 | Byte Offset: 0x158 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	DSI_RAW_DATA_BYTE_COUNT: Host RAW DATA byte count specifies the total number of bytes to send when "HOST_DSI_CONTROL[6] -> RAW_DATA" enabled

26.11.13 DSI_DSI_ULTRA_LOW_POWER_CONTROL_0

Ultra Low Power Sequence Control Register

Offset: 0x57 | Byte Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:8	0x0	DSI_ULTRA_LOW_POWER_DATA_LANE3: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
7:6	0x0	DSI_ULTRA_LOW_POWER_DATA_LANE2: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM

Bit	Reset	Description
5:4	0x0	DSI_ULTRA_LOW_POWER_DATA_LANE1: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
3:2	0x0	DSI_ULTRA_LOW_POWER_DATA_LANE0: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM
1:0	0x0	DSI_ULTRA_LOW_POWER_CLK_LANE: Ultra Low Power 0 = NORMAL 1 = ENTER_ULPM 2 = EXIT_ULPM

26.11.14 DSI_DSI_INIT_SEQ_DATA_8_0

DSI Init Sequence Write Data 8

Offset: 0x58 | Byte Offset: 0x160 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_8: DSI Init Sequence Write Data bits 31:0

26.11.15 DSI_DSI_INIT_SEQ_DATA_9_0

DSI Init Sequence Write Data 9

Offset: 0x59 | Byte Offset: 0x164 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_9: DSI Init Sequence Write Data bits 31:0

26.11.16 DSI_DSI_INIT_SEQ_DATA_10_0

DSI Init Sequence Write Data 10

Offset: 0x5a | Byte Offset: 0x168 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_10: DSI Init Sequence Write Data bits 31:0

26.11.17 DSI_DSI_INIT_SEQ_DATA_11_0

DSI Init Sequence Write Data 11

Offset: 0x5b | Byte Offset: 0x16c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_11: DSI Init Sequence Write Data bits 31:0

26.11.18 DSI_DSI_INIT_SEQ_DATA_12_0

DSI Init Sequence Write Data 12

Offset: 0x5c | Byte Offset: 0x170 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_12: DSI Init Sequence Write Data bits 31:0

26.11.19 DSI_DSI_INIT_SEQ_DATA_13_0

DSI Init Sequence Write Data 13

Offset: 0x5d | Byte Offset: 0x174 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_13: DSI Init Sequence Write Data bits 31:0

26.11.20 DSI_DSI_INIT_SEQ_DATA_14_0

DSI Init Sequence Write Data 14

Offset: 0x5e | Byte Offset: 0x178 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_14: DSI Init Sequence Write Data bits 31:0

26.11.21 DSI_DSI_INIT_SEQ_DATA_15_0

DSI Init Sequence Write Data 15

Offset: 0x5f | Byte Offset: 0x17c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_15: DSI Init Sequence Write Data bits 31:0

[THIS PAGE INTENTIONALLY LEFT BLANK]

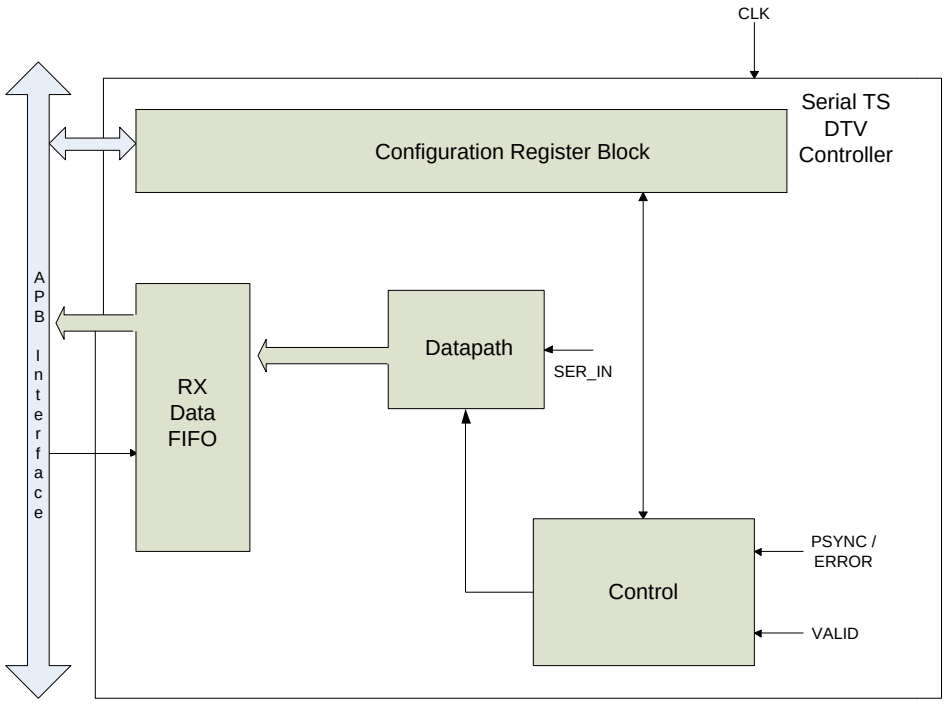
27.0 SERIAL TRANSPORT STREAM DTV CONTROLLER

The Tegra® 4 device supports a Serial Transport Stream (Serial TS) used to carry digital television data in some applications. The protocol is similar to the Motorola SPI protocol, but is not directly compatible. This section describes the Serial TS DTV controller.

27.1 Functional Description

The Serial TS DTV controller connects to the APB bus and works as a slave. It has a Receive FIFO size of 32 x 32 bits. Software programs the controller to generate transactions of a required packet length, where a transaction is a sequence of packets in the receive direction. The controller uses APB DMA to read from the FIFOs as required. Software reads the Data and FEC from the FIFO and does the necessary operation. The figure below shows the Serial TS DTV controller's internal blocks.

Figure 118: Transport Stream with Internal Blocks



Features

- Slave support for the serial TS interface for different mobile TV protocols
- Programmable polarity for valid / packet sync and error signals
- Minimum packet size is 4 bytes
- Capture of Reed-Solomon data
- 0-20 MHz frequency of operation
- Statistics counters
- DMA mastering capability

Hardware/Software Partitioning

- **Hardware** – Converts the serially incoming Data and FEC to parallel and stores them to make them visible to software. Updates the status registers (for different error conditions).
- **Software** – Reads the Data and FEC to perform necessary tasks, programs the control registers, and handles the status registers for different error conditions.

27.1.1 Signal Sanitization

The following internal signals provide flexibility to variations on the protocol:

Polarity normalized inputs:

- PSYNC = (TS_PSYNC ^ PSYNC_POL)
- VALID = (TS_VALID ^ VALID_ POL)
- ERROR = (TS_ERROR ^ ERROR_POL)

START: defined as one of the following (programming choice START_SELECT):

- Rising edge of VALID
- Rising edge of PSYNC
- Rising edge of (VALID & PSYNC)

START signal is internally used to delimit the beginning of the packet.

BODY_VALID: defined as one of the following (programming choice BODY_VALID_SELECT):

- byte_cnt = 0 on reset and gets incremented by 1 for each byte received. BODY_SIZE can be programmed to be in the inclusive range of 1-255 bytes (default is 188).
- The interval between START and the falling edge of VALID.

FEC_VALID: asserted from the deassertion of BODY_VALID and held high during FEC_SIZE bytes.

- FEC_SIZE can be programmed to be in the inclusive range of 0-127 bytes (default is 16).
- If FEC_SIZE = 0, this internal signal is not asserted.

PROTOCOL_SELECT: defined as one of the following (programming choice PROTOCOL_SELECT):

- 00b: ERROR and PSYNC are tied to zero
- 01b: ERROR is tied to the error/PSYNC input port. PSYNC is tied to zero
- 10b: ERROR is tied to zero. PSYNC is tied to the error/PSYNC input port.
- 11b: Reserved

CLK_MODE: defines one of the following with the programmable option of DTV_MODE_0.clk_mode:

- 0: a discontinuous clock source.
- 1: a continuous clock source.

A discontinuous clock source is one where the clock toggles exactly 8*N times for a given packet size of N bytes. Any number of clocks greater than 8*N is regarded as a continuous clock.

27.1.2 Protocol Handling

A serial interface packet consists of a BODY section and an FEC section.

BODY: TS packet. BODY is validated by BODY_VALID. The first byte of the packet is expected to be a sync byte.

FEC: FEC_SIZE bytes, FEC data (typically Reed-Solomon parity bytes). FEC is validated by FEC_VALID.

Packet capture stops after BODY and (if FEC_SIZE > 0) FEC are captured.

If START is reasserted prior to completing a full packet, the packet will be considered a PACKET_UNDERRUN_ERROR. The sections below describe error conditions.

27.1.3 Error Conditions

Error Packet

A packet is considered an error packet in any of the following cases:

1. **UPSTREAM_ERROR:** TS_ERROR is asserted during the first clock cycle of the packet and remains high during the capture of the whole packet.
2. **BODY_UNDERRUN_ERROR:** If the number of bytes captured in BODY (BODY SIZE) is less than EXPECTED BODY_SIZE. Occurs only when DTV_CONTROL_0.body_valid_select is set to 1.
3. **BODY_OVERRUN_ERROR:** If the number of bytes captured in BODY (BODY SIZE) is greater than EXPECTED BODY_SIZE. Occurs only when DTV_CONTROL_0.body_valid_select is set to 1.
4. **PACKET_UNDERRUN_ERROR:** None of the above (1 through 3) occur and If the total number of bytes captured (whole of the packet) is less than EXPECTED (BODY_SIZE + FEC_SIZE)

The error conditions described above are checked in the order specified. The first one encountered is reported. A packet will produce at most one error condition.

Statistics Counters

There are two statistics counters: ERROR_COUNTER and PACKET_COUNTER:

- Error packets are counted in ERROR_COUNTER.
- Total packets are counted in PACKET_COUNTER (regardless of whether they contain errors or not).
- Both statistics counters are sticky on 0xFFFF and are not free-running. Both are cleared on writes and cannot be preloaded with user-defined values.
- The byte stream stored in memory contains no explicit indication of the error.
- All packets are stored in memory regardless of their size and type.
- There is no specific hardware mechanism to signal software where the errors are located. Software is expected to perform the required checks to make sure the packet is correct.

Writing Error Packets into Memory

Error packets (including the case where they are signaled up front, i.e., UPSTREAM ERROR) are stored in memory with the captured length.

27.1.4 SERIAL TS Memory Format

Captured data is stored in memory according to the following format:

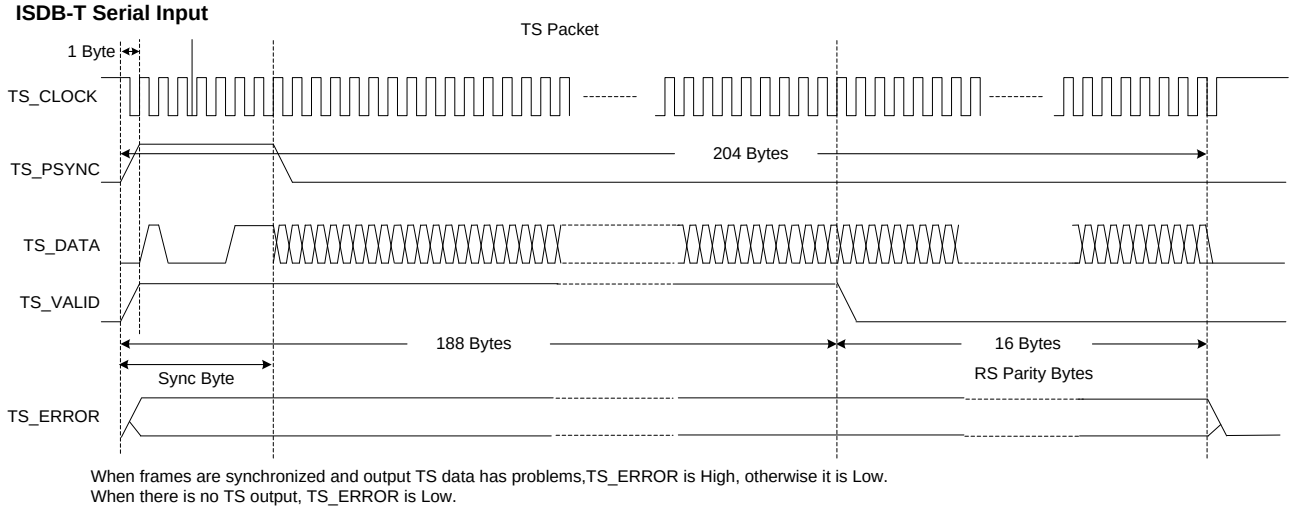
- BODY: BODY_SIZE bytes out of which the first byte is considered a sync byte and
- FEC: FEC_SIZE bytes, if FEC is present.

Functional Description

- The Start signal is generated as explained above with the programmable option START_SELECT.
- VALID is asserted for Body and deasserted for FEC, if at all it is taken into consideration with BODY_VALID_SELECT.
- VALID_POLARITY determines the polarity of the VALID signal.
- Sync byte is the first byte in body to indicate the start of a new packet. The value of the sync byte is fixed at 0x47 to indicate the new packet is properly aligned.
- Either Error/Sync or both are tied to zero based on the programmed value in PROTOCOL_SELECT. Error/Sync polarity depends on ERROR_POLARITY and PSYNC_POLARITY, respectively.
- Serially captured information on DATA is deserialized (shift register) and stored in a FIFO.

The timing diagram for a serial communication of a packet with DATA, PSYNC, and VALID is shown below:

Figure 119: Serial Communication Timing Diagram



When frames are synchronized and output TS data has problems,TS_ERROR is High, otherwise it is Low.
When there is no TS output, TS_ERROR is Low.

27.1.5 Control

The control logic includes:

- Generation of start signal and the polarity for PSYNC / ERROR and VALID
- Byte counter to count the number of bytes received in DATA and FEC
- Assert the error bits on respective error conditions
- End of packet generation based on the byte counter

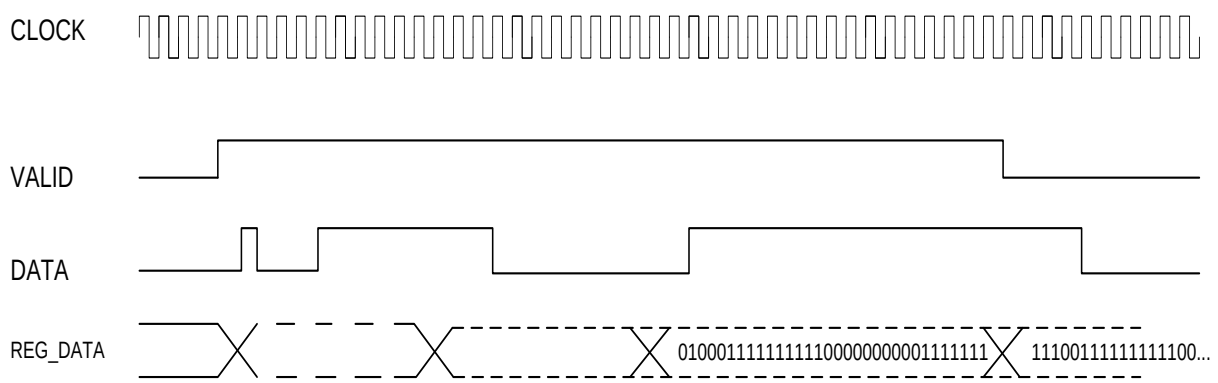
27.1.6 Datapath

Datapath logic performs these functions:

- Writes 32-bit data from the DTV clock domain into the sync FIFO
- Clears the status on writes to the respective error status bits
- Counts the number of packets (non-error and error)

The deserialization is shown in the following figure:

Figure 120: Deserializing of Incoming Serial Data



27.1.7 Status Information

There are certain bits of status information in the registers that the software can use to determine the status of the currently ongoing transfer.

RXF_OVF bit: This bit indicates an overflow in RX FIFO and is set to 1 whenever an error is detected during a receive operation.

RXF_FULL: This bit indicates whether the RX FIFO is full or not and is set to 1 when RX FIFO is full. If this bit is set to 1 during a reception, it indicates that the controller has received data in the RX FIFO and is waiting for the firmware to read these data before receiving any more data.

RXF_EMPTY: This bit indicates whether the RX FIFO is empty or not and is set to 1 when RX FIFO is empty.

INTERRUPT_STATUS: The respective bit is asserted high when one of the following occurs:

- Upstream error or
- Body Underrun error or
- Body Overrun error or
- Packet Underrun error.

The bit remains asserted until acted upon by the CPU to make it deasserted.

PACKET_COUNTER.VALUE: This field contains the number of packets that have been transferred during current transfer irrespective of whether they are error or not.

ERROR_COUNTER.VALUE: This field contains the number of error packets. Increments when there is an upstream error or when one of the error conditions is asserted.

27.2 Programming Guidelines

Follow these steps to program the Serial TS DTV controller:

1. Enable DTV in the DTV_SPI_CONTROL register
2. Program the DTV_MODE register for bit/byte swizzle, clock edge, etc.
3. Program the FEC size, body size, polarities, etc. in the DTV_CONTROL register
4. Enable DTV by writing a 1 to bit 0 of the DTV_MODE register

27.3 DTV Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

27.3.1 DTV_SPI_CONTROL_0

Offset: 0x40 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx1)

Bit	Reset	Description
0	0x1	SELECT: Kept to maintain compatibility. 1 = DTV enabled, 0 = SPI CTLR enabled 0 = DISABLED 1 = ENABLED

27.3.2 DTV_MODE_0

Mode Selection Register

DTV MODE Register por=0x00000002

Offset: 0x44 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6	0x0	BYTE_SWIZZLE: This determines byte order during deserialization. 0 = DISABLED 1 = ENABLED
5	0x0	BIT_SWIZZLE: This determines bit order during deserialization. 0 = DISABLED 1 = ENABLED
4	0x0	CLK_EDGE_SELECT: This determines the clock edge to be used to sample DTV input signals. 0 = POSEDGE 1 = NEGEDGE
3:2	0x0	PROTOCOL_SELECT: For selecting the pin configuration for VD[1]. NONE: ERROR is tied to 0. PSYNC is tied to 0. ERROR: ERROR is tied to VD[1]. PSYNC is tied to 0. PSYNC: ERROR is tied to 0. PSYNC is tied to VD[1]. 0 = NONE 1 = ERROR 2 = PSYNC 3 = RESERVED
1	0x0	CLK_MODE: Determines if the input clock is continuous or discontinuous. 0 = DISCONTINUOUS 1 = CONTINUOUS
0	0x0	ENABLE: DTV protocol handling global enable. 0 = DISABLED 1 = ENABLED

27.3.3 DTV_CONTROL_0

Mode Control Register

DTV CONTROL Register por=0x00000000

Offset: 0x48 | Read/Write: R/W | Reset: 0x10bc0300 (0bx0010000101111100xx000011x000x000)

Bit	Reset	Description
30:24	0x10	FEC_SIZE: Stores the number of FEC bytes to capture, after the BODY has been captured.
23:16	0xbc	BODY_SIZE: Stores the number of BODY bytes to capture including PSYNC.
13:8	0x3	RX_FIFO_ATTN_LEVEL: Receive FIFO trigger level. 0x0 = 1 word DMA trigger is asserted when at least one word full in the FIFO. 0x1 = 2 word DMA trigger is asserted when at least 2 words are full in the FIFO
6	0x0	BODY_VALID_SELECT: Determines if VALID is used during BODY packet capture. IGNORE: VALID signal is ignored during the BODY capture. GATE: VALID gates the capture of BODY data. 0 = IGNORE 1 = GATE
5:4	0x0	START_SELECT: Determines the START of the packet condition. PSYNC: PSYNC assertion rising edge. VALID: VALID assertion rising edge. BOTH: PSYNC && VALID asserted rising edge. 0 = RESERVED 1 = PSYNC 2 = VALID 3 = BOTH
2	0x0	ERROR_POLARITY: Indicates the polarity of ERROR, has effect only when MODE.ENABLE == ENABLED. LOW: Indicates the polarity of ERROR is active low. HIGH: Indicates the polarity of ERROR is active high. 0 = HIGH 1 = LOW
1	0x0	PSYNC_POLARITY: Indicates the polarity of PSYNC, has effect only when MODE.ENABLE == ENABLED. LOW: Indicates the polarity of PSYNC is active low. HIGH: Indicates the polarity of PSYNC is active high. 0 = HIGH 1 = LOW
0	0x0	VALID_POLARITY: Indicates the polarity of VALID, has effect only when MODE.ENABLE == ENABLED. LOW: Indicates the polarity of VALID is active low. HIGH: Indicates the polarity of VALID is active high. 0 = HIGH 1 = LOW

27.3.4 DTV_PACKET_COUNT_0

Packet Count Register

DTV PACKET COUNT Register por=0x00000000

Offset: 0x4c │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	PACKET_COUNT_VALUE: Holds the current value of the received packet counter. This counter increments in the presence of a new packet, regardless of whether it is a flagged error or not. The counter is cleared on writes, and it cannot be loaded with a preloaded value.

27.3.5 DTV_ERROR_COUNT_0

Error Count Register

DTV ERROR COUNT Register por=0x00000000

Offset: 0x50 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	ERROR_COUNT_VALUE: Holds the current value of the error packet counter. This counter increments in the scenario when a packet is flagged as an error or when a protocol violation has occurred. The counter is cleared on writes, and it cannot be loaded with a preloaded value This register returns interrupt status when read. When this register is written, the interrupt status corresponding to the bits written with 1 will be reset. The interrupt status corresponding to the bits written with 0 will be left unchanged.

27.3.6 DTV_INTERRUPT_STATUS_0

Interrupt Enable

DTV INTERRUPT STATUS Register por=0x00000000

Offset: 0x54 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	PACKET_UNDERRUN_ERROR_INT_STATUS: Start condition detected prior to receiving a full packet. 0 = Interrupt not detected. 1 = Interrupt detected 0 = NOINTR 1 = INTR
2	0x0	BODY_OVERRUN_ERROR_INT_STATUS: This bit shows the status of the condition when the DTV input gets a body overrun error, i.e., more bytes in body than specified. 0 = Interrupt not detected. 1 = Interrupt detected 0 = NOINTR 1 = INTR
1	0x0	BODY_UNDERRUN_ERROR_INT_STATUS: This bit shows the status of the condition when the DTV input gets a body underrun error. 0 = Interrupt not detected. 1 = Interrupt detected 0 = NOINTR 1 = INTR
0	0x0	UPSTREAM_ERROR_INT_STATUS: This bit shows the status of the condition when the DTV input gets an upstream error. 0 = Interrupt not detected. 1 = Interrupt detected 0 = NOINTR 1 = INTR

27.3.7 DTV_STATUS_0

DTV Status Register por=0b0xx

Offset: 0x58 │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xx)

Bit	R/W	Reset	Description
2	RW	0x0	RXF_UNR: RX FIFO Underrun: This bit is set to 1 whenever software tries to read from an empty RX FIFO. An interrupt is generated if the interrupt enable is set for receive operations. Software writes a 1 to clear this bit. Clearing this bit also clears the interrupt.
1	RO	X	RXF_EMPTY: RX FIFO empty status. Hardware sets this bit to 1 if the RX FIFO is empty. Otherwise this bit is set to 0. 0 = NOT_EMPTY 1 = EMPTY
0	RO	X	RXF_FULL: RX FIFO full status. Hardware sets this bit to 1 if the RX FIFO is full. Otherwise, this bit is set to 0. 0 = NOT_FULL 1 = FULL

27.3.8 DTV_RX_FIFO_0

DTV RX FIFO Register por=0x00000000

Offset: 0x5c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RD_DATA: RX FIFO

[THIS PAGE INTENTIONALLY LEFT BLANK]

28.0 HIGH-DEFINITION MULTIMEDIA INTERFACE

The High-Definition Multimedia Interface (HDMI) receives pixels from the Display Controller and audio from the HDA. It combines and transmits them together.

28.1 Features

- 300 MHz pixel clock supported to enable 1080p @ 60 3D and 4k x 2k @ 30 resolutions
- HARDWARE_N feature enabled
- New TMDS I/O macro with new controls
- S/PDIF audio is not supported
- HDMI in power-gated partition DISB (with displayB)
- Display CMU added, allowing color management
- HDMI and display have more limited clocking options. HDMI always uses PLLD2.
- Audio features for HD audio include:
    - 2, 6, or 8 channels of LPCM (i.e., uncompressed linear PCM)
    - 16, 20, or 24 bits per sample
    - 32, 44.1, 48, 88.2, 96, 176.4, 192 K samples/second

28.2 Programming Guidelines

The overall sequence to start up HDMI is:

- Program power, pin mux, clocks, resets, etc.
- Copy HDCP keys from KFUSE to HDMI
- Program display timing registers, etc.
- Program HDMI registers and Serial Output Resource (SOR) sequencer
- Start HDMI SOR
- Start display

Since audio is entirely asynchronous to HDMI or the display, the HDA module can be started any time.

28.2.1 Power

Turn on the necessary power supplies.

- MIPI_PLL
- HDMI_PLL_HVDD
- HDMI_AVDD: 3.3V
- +5V for DDC, HDMI_INT, CEC, etc.

These need to be enabled in the PMIC via I^2C.

28.2.2 DDC / I2C

DDC I^2C is used for E-EDID reading, HDCP, and other communication with downstream HDMI device. I^2C has no connection to HDMI block; the platform defines which I^2C engine is connected to the DDC.

28.2.3 Hot Plug

HDMI_INT is a pad that feeds the GPIO block and the PMC. It has no connection to the HDMI block. It operates as a general-purpose I/O pin.

28.2.4 Clocks

HDMI uses two main clocks:

- hdmi_clk: variable from 25.2 MHz to 300 MHz based on video pixel rate. Must be identical frequency and source as display's pixel clock. (i.e., display and HDMI clocks are isochronous).
- hda2codec_2x_clk: fixed 48 MHz HD Audio bit clock. Same source as HDA Controller's bitclk. The HDA Controller clock enable should not affect it.

And three others:

- display2hdmi_pclk and displayb2hdmi_pclk: Copy of display pixel clocks for display2hdmi and displayb2hdmi pixel buses. Only used to write display data into FIFOs.
- host1x2hdmi_clk: Copy of host1x clock for host interface; only used to write host1x transactions into the FIFO.

All three main clocks have common first-level gate: the CAR CLK_ENB_HDMI register. hdmi_audio_clk has a second level gate controlled by hardware – it is gated when S/PDIF is not used.

The display controllers and HDMI have separate clock source registers. However, they must be programmed to the same source and the same frequency. Typical settings for a 13 MHz oscillator using PLLD2 are listed in the following table:

Pixel Clk MHz	PLLD2 Frequency	PLL M	PLL N	HDMI_ CLK_ DIVISOR	Display SHIFT_ CLK_ DIVIDER	HDMI_ CLK_ SRC	DISP1_ CLK_ SRC
27	216/2	13	216	0x6 (/4)	0x6 (/4)	PLLD2_OUT0	PLLD2_OUT0
74.25	594/2	13	594	0x6 (/4)	0x6 (/4)	PLLD2_OUT0	PLLD2_OUT0
148.5	594/2	13	594	0x2 (/2)	0x2 (/2)	PLLD2_OUT0	PLLD2_OUT0
297	594/2	13	594	0x0 (/1)	0x0 (/1)	PLLD2_OUT0	PLLD2_OUT0

In addition, the HDA modules must use the same oscillator as display and HDMI, since audio and video clock frequencies must be a fixed ratio. For HDA Audio, HDA2CODEC_2X clock must be enabled and configured for 48 MHz.

28.2.5 Display Timing

Frame timing is controlled by the display. HDMI has certain requirements, however, imposed by both the module itself and the HDMI standards.

HDMI gets the following signals from the display:

- h_blank – derived in hardware from h_ref_to_sync, h_sync_width, h_back_porch, h_disp_active, h_front_porch
- v_blank – derived in hardware from v_ref_to_sync, v_sync_width, v_back_porch, v_disp_active, v_front_porch
- h_sync – derived from h_ref_to_sync, h_sync_width
- v_sync – derived from v_ref_to_sync, v_sync_width
- video_preamble – derived from h_pulse2_control, h_pulse2_position.

EIA (CEA-861-B) Formats	ref_to_sync	sync_width	back_porch	disp_active	front_porch
480P_60 (2,3)					
horizontal	1	62	60	720	16
vertical	1	6	30	480	9
720P_60 (4)					
horizontal	1	40	220	1280	110
vertical	1	5	20	720	5
720P_50 (19)					
horizontal	1	40	220	1280	440
vertical	1	5	20	720	5
640x480_60 (1)					
horizontal	1	96	48	640	16
vertical	1	2	33	480	10
576p_50 (17/18)					
horizontal	1	64	68	720	12
vertical	1	5	39	576	5
1080p_60 (16)					
horizontal	1	44	148	1920	88
vertical	1	5	36	1080	4
4kx2k_30					
horizontal	1	88	296	3840	176
vertical	1	10	72	2160	8

The video_preamble uses h_pulse2, with the following settings:

disp_signal_options0.h_pulse2_enable	ENABLE
h_pulse2_control.h_pulse2_mode	NORMAL
h_pulse2_control.h_pulse2_polarity	HIGH
h_pulse2_control.h_pulse2_v_qual	VACTIVE
h_pulse2_control.h_pulse2_last_end	END_A
h_pulse2_position.h_pulse2_start_a	h_ref_to_sync + h_sync_width + h_back_porch - 10
h_pulse2_position.h_pulse2_end_a	h_pulse2_start_a + 8

Miscellaneous registers required:

DISP_TIMING_OPTIONS.VSYNC_H_POSITION	1
DISP_CLOCK_CONTROL.PIXEL_CLK_DIVIDER	PCD1
DISP_COLOR_CONTROL.DITHER_CONTROL	DISABLE
DISP_WIN_OPTIONS.HDMI_ENABLE	ENABLE

28.2.6 Display Datapath

28.2.6.1 Dither

Dithering should be disabled, since HDMI uses a full 8 bits per component.

28.2.6.2 Component Range

For most HDMI resolutions (i.e., everything except VGA 640x480), R/G/B should be scaled to lie in the [16, 235] nominal range; See the HDMI specification, section 6.6. The range scaling can be done in either of two places in Tegra® 4 series processors; CMU is recommended:

- Front of pipe (per-window), using color palette. Palette can contain function int(i/255 * 219 + 0.5) + 16 where ‘i’ is palette index 0..255.
- After compositing, in the CMU block (recommended).

```
proc cmu_lut1_flt2fix {flt} {
    set fx [expr floor($flt * 4095.0 + 0.5)]
    if {$fx > 4095} {
        error "cmu_lut1_flt2fix out of range input $flt"
    }
    return $fx
}
proc cmu_lut2_flt2fix {flt} {
    set fx [expr floor($flt * 255.0 + 0.5)]
    if {$fx > 255} {
        error "cmu_lut1_flt2fix out of range input $flt"
    }
    return $fx
}
proc srgb_ungamma {x args} {
    if {$x <= 0.03928} {
        set y [expr $x/12.92]
    } else {
        set y [expr pow(($x + 0.055)/1.055, 2.4)]
    }
    return [cmu_lut1_flt2fix $y]
}
proc rec709_gamma_offset16 {x args} {
    if {$x <= 0.018} {
        set y [expr $x * 4.5]
    } else {
        set y [expr 1.099 * pow($x, 0.45) - 0.099]
    }
    set lut2 [expr 16 + [cmu_lut2_flt2fix $y]]
    return [expr min(255, max($lut2, 0))]
}

#
# program CMU to convert to limited-range RGB with Rec. 709 gamma.
# Display pipe should be using full-range RGB.
# [0..255] is mapped to [16,235].
# NOTE: sRGB primaries and whitebpoint are same as 709, so matrix is just scale.
# We could do everything in LUT as well.
#
proc cmu_to_rec709_rgb219 {} {
```

```
    set scale [expr 219.0 / 255.0]
    set mat "
        $scale 0.0    0.0    \
        0.0    $scale 0.0    \
        0.0    0.0    $scale \
    "
    cmu lut1 srgb ungamma
    cmu csc $mat
    cmu lut2 rec709 gamma offset16
    DISP COLOR CONTROL.CMU ENABLE ENABLE
}
```

For most HDMI modes, codes 0 and 255 must be removed; HDMI'S NV_PDISP_INPUT_CONTROL.ARM_VIDEO_RANGE = LIMITED enables this.

28.2.6.3 Gamma

Most content uses sRGB gamma which is not quite the same as Rec. 709 gamma used by HDMI for most modes. Post-composite gamma can be changed in the CMU. The above code illustrates these.HDMI basics.

28.2.7 HDMI Basics

- Select display a/b source with NV_PDISP_HDMI_SRC_SELECT
- If appropriate to the platform, program HDCP timing dividers, based on the HDMI pixel clock. For Tegra 4 platforms, ROM I^2C clock frequency should be 200 KHz.
- Program SOR reference clock, based on the HDMI pixel clock – see the NV_PDISP_SOR_REFCLK register description in this section. For example,

```
dispclk div 8 2 = int(dispclk freq / 1000000.0 * 4)
NV_PDISP_SOR_REFCLK.DIV_INT = dispclk_div_8_2 >> 2
NV_PDISP_SOR_REFCLK.DIV_FRAC = dispclk_div_8_2 & 0x3
```

- Program SOR sequences for power up and power down.

28.2.8 TMDS Macro Configuration

The default TMDS macro settings for Tegra 4 devices are listed in the table below:

HDMI Register	HDMI Register init	Default	Clamp Value (HDMI)
BG_TEMP_COEF	3	3	3
BG_VREF_LEVEL	4	4	4
DIV_RATIO_OVERRIDE	0		0
n/a (PMC)			
FUSE_OVERRIDE	0	n/a	n/a
NV_PDISP_SOR_IO_PEAK_CURRENT	0	0	0
NV_PDISP_SOR_LANE_DRIVE_CURRENT	0x20	0x18	0x20
KICKSTART	0	0	0
KVCO_2NDVCO	0	0	0
NV_PDISP_SOR_PLL1.LOADADJ	3	3	3
LOADADJ_BYPN	0	0	0
LOADADJ_SYNC_EN	0	0	0
NV_PDISP_SOR_PLL0.PDBG	1		1
SOR_PLL0.PWR	1		1
PDPORT	1		1

HDMI Register	HDMI Register init	Default	Clamp Value (HDMI)
SOR_CSTM.PD_TX*/etc.	0xf		0xf
NV_PDISP_PE_CURRENT	8	8	8
NV_PDISP_SOR_PLL1.PE_EN	0		0
PLL_BYPASS	0	0	0
PLL_NDIV_RATIO	0	NO USE	0
PLL_PDIV_RATIO	0	1	0
ICHPMP	2	2	2
FILTER	0	0	0
VCOCAP	3		3
PULLDOWN	1	NO USE	1
AVDD10_LEVEL	2	2	2
AVDD10_LOAD	3	3	3
AVDD23_LEVEL	2	2	2
AVDD23_LOAD	3	3	3
REG_BYPASS	0	0	0
RESISTORSEL	1	1	1
SOR_PLL2.AUX	0	0	0
TMDS_TERMADJ	9	6	9
TMDS_TERM	0		0
TESTMUX	0	0	0
TX_REG_LOAD	0	NOT USED	0
VCOCALIB_ENB	0	0	0
VCOCALIB_OVRWRB	0	0	0
VCOCALIB_TS0	0	0	0
VCOLIMIT_DISABLE	0	0	0
VCOLIMIT_SEL	0	0	0

Note: NV_PDISP_SOR_LANE_DRIVE_CURRENT. LANE0..3 are only used if NV_PDISP_SOR_LANE_DRIVE_CURRENT.FUSE_OVERRIDE is TRUE. HDMI TMDS calibration can be performed and burned into fuses, but are not currently .

28.2.9 HDA Controller Initialization

Basic operations:

- HDA IPFS initialization, FPCI and HDA configuration, and memory spaces allocated
- Deassert soft reset, bring up link. AZA.GCTL.CRST = DEASSERT, and wait for reset completion by spinning until CRST = DEASSERT. Wait 4 frame times (poll AZA.WALCLK for 2000 ticks to elapse)
- Verify internal codec (and external if desired) have been detected by reading AZA.WAKEEN.SDIWAKE3 for internal codec, and SDIWAKE1 for external.

28.2.10 HDA Audio Codec Initialization

When HDA is the audio source, the codec must be initialized before audio can flow. The internal codec uses industry-standard verbs, so in theory, the out-of-box driver should correctly discover and configure the codec. See the HD Audio specification. Below is a summary of the necessary verbs:

1. AOC widget: SET_CONV_STREAM_CHAN with desired output stream ID
2. AOC widget: SET_CONV_FMT to desired audio format (sampling frequency, sample depth, number of channels)
3. AOC widget: SET_DIGITAL_CONV_CONTROL1
   - Bit 0: 1 (DigEn=1)
   - Bit 3..6: channel status pre/copy/audio/pro bits
   - Bit 7: channel status CC bit 7 (called GEN_LEVEL)
4. AOC widget: SET_DIGITAL_CONV_CONTROL2:
   - Bit 6:0: channel status CC bits 6:0
5. Pin widget: SET_CONVERTER_CHANNEL_TO_DIGITAL_SLOT_MAPPING= 0x22 and SET_CONVERTER_CHANNEL_TO_DIGITAL_SLOT_MAPPING= 0x33 to undo default swap of channels 2 and 3
6. AOC widget: SET_STRIPE_CONTROL to select number of lanes. Recommend programming to 2 lanes all the time.
7. Pin widget: SET_PIN_WIDGET_CTRL = 0x40, to enable output and use PCM packets.
8. If desired to use HDA for info packets, program them with SET_DIP_INDEX/SET_DIP_DATA and enable with SET_DIP_XMIT_CTRL.

28.2.11 Audio

Two methods of generating the CTS portion of ACR packets are supported:

- In “Force SW CTS”, software computes CTS for the current audio and pixel clocks and writes it to registers.
- In “Hardware CTS”, hardware computes CTS automatically.

Likewise, two methods for generating N portion are supported.

- In “SW N”, software programs current desired N into 0441_SUBPACK_HIGH register and AUDIO_N.VALUE.
- In “HW N”, software programs all the ACR_*_SUBPACK_HIGH per-frequency N, then hardware chooses the one to use.

1. Set NV_PDISP_HDMI_ACR_CTRL (all fields) to 0
2. Reset N counter: HDMI.NV_PDISP_AUDIO_N.RESETF = ASSERT, GENERATE=ALTERNATE, LOOKUP per table
3. Based on pixel clock and audio sample clock frequencies, program the appropriate ACR N value into NV_PDISP_HDMI_ACR_0441_SUBPACK_ HIGH (see HDMI 1.2a specification, section 7.2.3). LSB of N goes in SB6, and MSB into SB4.
4. Likewise, program the appropriate ACR CTS value into NV_PDISP_HDMI_ACR_0441_SUBPACK_LOW. The LSB of CTS goes in SB3, and the MSB into SB1.
5. Program the computed AVAL into the appropriate SOR_AUDIO_AVAL_ register
6. Set NV_PDISP_AUDIO_N.VALUE = N – 1 (from above)
7. Set NV_PDISP_HDMI_ACR_*_SUBPACK_HIGH.ENABLE = YES.
8. Set NV_PDISP_HDMI_SPARE = 0x10000 OR’d with bits 0 and 1
9. Bring the N counter of reset: set HDMI.NV_PDISP_AUDIO_N.RESETF = DEASSERT
10. For S/PDIF, for each supported audio sample frequency, program NV_PDISP_AUDIO_FS*.HIGH and LOW:
    - eight_half = int(8 * 216000000 / (audio_sample_freq * 128)). For example, for 48 kHz audio: eight_half = 8 * 216000000 / (48000 * 128) = 281.25

- HIGH = eight_half + 9
- LOW = eight_half - 9

11. Program the audio source with SOR_AUDIO_CNTRL0.SOURCE_SELECT = SPDIF for S/PDIF, HDAL for HDA, or AUTO. AUTO automatically selects HDA if HDMI's internal HDA codec has been enumerated by HDA controller, else SPDIF is selected.
12. Enable HDMI audio: Set HDMI_CTRL.ENABLE = EN, HDMI_GENERIC_CTRL.AUDIO = EN, AUDIO_ENGINE.PWRDWN=0

Note:	CTS and N values depend on the ratio of the actual pixel clock and audio clock rates. The chip's clock generators cannot create every frequency exactly. For example 25.2 MHz (for 640x480p@60) cannot be exactly generated – 25.25 is the result. Thus CTS and N must be adjusted accordingly.

28.2.12 Other HDMI Configurations

1. Set HDMI_CTRL.MAX_AC_PACKET based on horizontal blanking width. Using display timing registers, MAX_AC_PACKET = floor((H_SYNC_WIDTH + H_BACK_PORCH + H_FRONT_PORCH – HDMI_CTRL.REKEY – 18) / 32)
2. Program and enable Audio InfoFrames. When using HDA audio, its contents come from verbs sent to internal codec, so this step can be skipped. When using SPDIF audio, Audio InfoFrame contents come from HDMI registers:
   - Set HDMI_AUDIO_INFOFRAME_HEADER = 0x000A0184
   - Set PB4-7 = 0, typically, in HDMI_AUDIO_INFOFRAME_SUBPACK0_HIGH
   - Set PB1-3 and CRC in HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW. Typically PB1 = 1, and PB2 and PB3 are 0. CRC = (256 – (0x0a + 0x01 + 0x84 + PB1 + PB2 …) & 0xff) & 0xff
   - Set HDMI_AUDIO_INFOFRAME_CTRL.ENABLE = EN
3. Program and enable AVI InfoFrames. When using HDA Audio, this *may* come from the HDA Codec if desired, using one of its Generic packets. If so, do not enable the HDMI one:
   - Set HDMI_AVI_INFOFRAME_HEADER = 0x000D0282
   - Program checksum, PB1, PB2, PB3 in HDMI_AVI_INFOFRAME_SUBPACK0_LOW. PB4 contains the video format code from the CEA-861-D specification. See HDMI and EIA/CEA-861-D for all the fields. CRC is computed over header and PB1…PB13 as above.
   - Program PB4…PB6 in HDMI_AVI_INFOFRAME_SUBPACK0_HIGH, and PB7..PB13 in HDMI_AVI_INFOFRAME_SUBPACK1_LOW / HIGH.
   - Set HDMI_AVI_INFOFRAME_CTRL.ENABLE = EN
4. For HDMI 1.4 3D, program and enable the GENERIC InfoFrame to send an HDMI Vendor-Specific InfoFrame. When using HDA Audio, this *might* come from the HDA Codec, if desired, using one of its Generic packets. If so, do not enable the HDMI one.
   - Set HDMI_GENERIC_HEADER = 0x00LL0181 where LL is length of packet – 0x05 for the basic version described here.
   - HDMI_GENERIC_SUBPACK0_LOW.PB1/PB2/PB3 = 0x03/0c/00 (IEEE registration ID)
   - PB4 = 0x40 (Hdmi_video_format = 3d_structure present)
   - PB5 = 0x00 (3D_Structure = Frame packing)
   - PB6...PB27 = 0 (no 3D_Ext_Data for Frame packing)
   - Compute checksum over header and PB1..PB27 and put in PB0, as above.
   - HDMI_GENERIC_CTRL.ENABLE=EN. OTHER, SINGLE, HBLANK= DIS.
   - Be sure to keep HDMI_GENERIC_CTRL.AUDIO = EN

28.2.13 Start HDMI SOR

1. Enable TMDS macro with SOR_PLL0.PWR=0, VCOPD=0, PULLDOWN=0. Observe the power sequencing requirement. After PWR is cleared, you must wait 10 microseconds before setting PDBG=0.
2. Write SOR_PWR with NORMAL_STATE = PU, SAFE_STATE = PD, SETTING_NEW = TRIGGER.
3. Write SOR_PWR with the same settings but with SETTING_NEW = DONE.
4. Poll SOR_PWR for SETTING_NEW = DONE.
5. Write SOR_STATE2 with ASY_OWNER=HEAD0, SUBOWNER=3, ASY_PROTOCOL=SINGLE_TMDS_A, ASY_HSYNCPOL=POSITIVE_TRUE, ASY_VSYNCPOL=POSITIVE_TRUE, ASY_DEPOL=POSITIVE_TRUE.
6. Write SOR_STATE1 with ASY_HEAD_OPMODE=AWAKE, ASY_ORMODE=NORMAL, ATTACHED=0, ARM_SHOW_VGA=0.
7. Write SOR_STATE0 with UPDATE=0.
8. Write SOR_STATE0 with UPDATE=1.
9. Write SOR_STATE1 with ATTACHED=1.
10. Write SOR_STATE0 with UPDATE=0.

28.2.14 Start Display

- Set DISP_WIN_OPTIONS.HDMI_ENABLE = ENABLE.
- Start the display with DISPLAY_COMMAND.DISPLAY_CTRL_MODE and DISPLAY_POWER_CONTROL.

28.3 HDCP

HDCP requires the presence of a correctly programmed KFUSE. Two main sequences are downstream authentication and upstream authentication. Downstream authentication permits encryption of the HDMI output, and involves a handshake between the HDMI module and the display device. Upstream authentication is more of a software convention between the application and HDMI driver, although the HDMI module does perform KFUSE operations to assist.

Communication with the downstream receiver uses DDC, which requires I^2C.

28.3.1 HDCP KFUSE Key Copying

HDCP keys are stored encrypted in the KFUSE module. Before starting HDCP, software must copy the keys (576 bytes) from the KFUSE to the HDMI registers.

1. To enable use of KFUSE keys, set NV_PDISP_KEY_CTRL.LOCAL_KEYS = ENABLED.
2. Bring the KFUSE module out of reset and enable the clock, if not already enabled. This can be before or after HDMI is out of reset.
3. Wait for the KFUSE to read the keys from the fuse to its SRAM and perform error correction and CRC checking:
   a. Poll for KFUSE.STATE.DONE == 1
   b. Check KFUSE.STATE.CRCPASS == 1
   c. Check that NV_PDISP_KEY_CTRL.PKEY_LOADED == TRUE
4. Prepare to copy:
   a. KFUSE.KEYADDR.ADDR = 0
   b. KFUSE.KEYADDR.AUTOINC = 1
5. For each line in 0 .. 35 # 576 bytes = 144 words / 4 words-per-line = 36 lines
   a. For each word in 0..3:

      data[word] = KFUSE.KEYS # auto-increments read address

b. HDMI.NV_PDISP_KEY_HDCP_KEY_0 = data[0]
c. HDMI.NV_PDISP_KEY_HDCP_KEY_1 = data[1]
d. HDMI.NV_PDISP_KEY_HDCP_KEY_2 = data[2]
e. HDMI.NV_PDISP_KEY_HDCP_KEY_3 = data[3]
f. HDMI.NV_PDISP_KEY_HDCP_KEY_TRIG.LOAD_HDCP_KEY = TRIGGER # trigger decryption of 16 bytes
g. If line == 0: HDMI.NV_PDISP_KEY_CTRL.AUTOINC = DISABLED. Else: HDMI.NV_PDISP_KEY_CTRL.AUTOINC = ENABLED
h. HDMI. NV_PDISP_KEY_CTRL.WRITE16 = TRIGGER # wait for decrypt, and copy decrypted 16 bytes to the local key store.
i. Poll for HDMI. NV_PDISP_KEY_CTRL.WRITE16 == DONE

6. Write NV_PDISP_KEY_SKEY_INDEX.IDX_VALUE with the AES key selector that was used to encrypt the keyglob data. This is currently fixed at 0.

28.3.2 Downstream Authentication

This can be done before enabling transmission.

1. Set RG_HDCP_CTRL.ONEONE = ENABLE. RG_HDCP_CTRL.RUN = YES. Causes the RTL to access HDCP keys and compute An and other parameters.
2. Poll for RG_HDCP_CTRL.AN = VALID
3. Read Aksv from RG_HDCP_AKSV_LSB/MSB and check validity (exactly 20 of 40 bits = 1). If invalid, the KFUSE might be blank.
4. Read Bksv from the receiver (via I2C), check validity (20 high bits), and write it into RG_HDCP_BKSV_LSB/MSB.
5. Read An from RG_HDCP_AN_LSB/MSB, and write to the receiver via I^2C.
6. Read Aksv from RG_HDCP_AKSV_LSB/MSB, check validity (20 high bits), and write to receiver via I^2C.
7. Poll for RG_HDC_CTRL.R0 = VALID
8. Read RG_HDCP_RI and compare with the receiver's RI. They should match.

If the above procedure is successful, then RG_HDCP_CTRL.CRYPT can be set to ENABLE.

To verify that encryption is enabled, check that RG_HDCP_CTRL.RUN == YES.

28.3.3 Upstream Authentication

The HDMI module does not do the protocol itself. Rather, it provides primitives that software can use. These primitives access HDCP keys and status, and compute values.

Typically, downstream authentication has been completed before doing the upstream.

The "Upstream Setup" sequence is:

1. Set RG_HDCP_CMODE.MODE = READ_S
2. Poll for RG_HDCP_CTRL.SPRIME = INVALID
3. Program RG_HDCP_CN_LSB/MSB and RG_HDCP_CKSV_LSB/MSB with Cn and Cksv. NOTE: CKSV_MSB must be last since it is a trigger.
4. Poll for HDCP_CTRL.SPRIME = VALID
5. Read Dksv from RG_HDCP_DKSV_LSB/MSB and verify validity
6. Read Cs from RG_HDCP_CS_LSB/MSB
7. Read S' from RG_HDCP_SPRIME_LSB1/LSB2/MSB

Software uses these read values to perform the authentication handshake with application software.

The “Upstream M” sequence is:

1. Set RG_HDCP_CMODE.MODE = READ_M
2. Poll for RG_HDCP_CTRL.MPRIME = INVALID
3. Program RG_HDCP_CN_LSB/MSB and RG_HDCP_CKSV_LSB/MSB with Cn and Cksv. NOTE: CKSV_MSB must be last since it is a trigger.
4. Poll for HDCP_CTRL.MPRIME = VALID
5. Read Dksv from RG_HDCP_DKSV_LSB/MSB and verify validity
6. Read M' from RG_HDCP_MPRIME_LSB/MSB

28.4 Audio / Display Driver Communication

HDMI combines video and audio and so the display and audio drivers need to coordinate. The HDA Codec and HDMI implement the methods defined in the audio specifications.

28.4.1 Hot-Plug and ELD (EDID-like data)

When the HDMI driver receives notification of a hot-plug interrupt, it reads the downstream TV’s E-EDID ROM. The EDID contains audio-related data, so the audio driver needs it, too. The HDMI can send the hot-plug event and EDID buffer to the Audio driver via the HDA Codec.

Note:	See register comments in SOR_AUDIO_HDA_PRESENSE and SOR_AUDIO_HDA_ELD_BUFWR later in this section.

28.4.2 Copy-Protection

The audio driver can query copy-protection state (i.e., HDCP enabled), and request that HDCP be enabled by the HDMI driver.

Note:	See SOR_AUDIO_HDA_CP register and CP_REQUEST interrupt in INT_STATUS later in this section.

28.4.3 InfoPacket

In HDA mode, the Audio InfoFrame data is provided by the internal HDA Codec, not by the HDMI_AUDIO_INFOFRAME* registers. The SET_DIP_INDEX/SET_DIP_DATA verbs write this data.

Three generic infopackets are provided by the HDA Codec, in addition to the exiting generic packet.

28.4.4 Miscellaneous

For Audio to HDMI signaling, the HDA Codec has SCRATCH verbs that are reflected in the HDMI SOR_AUDIO_HDA_CODEC_SCRATCH* registers and CODEC_SCRATCH* interrupts.

For HDMI to Audio signaling, HDMI SOR_AUDIO_HDA_SCRATCH* registers, when written, are reflected to the HDA Codec GET_NV_SCRATCH_REGN_FROM_DISP verb and cause an Unsolicited Response (interrupt in the HDA Controller).

28.5 Clock Use Cases

Tegra 4 HDMI has the following clock sources. All five are required for HDMI operation.

DISP1 or DISP2	Display controller (pixel clock)
HDMI	HDMI controller (pixel clock)
HDA2CODEC_2X	HDA interface clock (48 MHz)
HDA2HDMICODEC	HDMI audio clock
HDA	HDA controller

28.6 CTS/N/AVAL Algorithm

```
void get_hda_cts_n(
// inputs:
NvU32 audio_freq_hz /* Fs */,
NvU32 pixclk_freq_hz /* Fpix */,
// outputs:
NvU32 &best_cts, NvU32 &best_n, NvU32 &best_a)
{
    // Ideal ACR interval is 1000 hz (1 ms);
    // acceptable is 300 hz .. 1500 hz
    const int min_n = 128 * audio_freq_hz / 1500;
    const int max_n = 128 * audio_freq_hz / 300;
    const int ideal_n = 128 * audio_freq_hz / 1000;

    float min_err = 100.0;

    best_n = 0;
    best_cts = 0;
    best_a = 0;

    for (int n = min_n; n <= max_n; n ++)
    {
        float cts_f = pixclk_freq_hz * (float)n / (128.0 * (float)audio_freq_hz);
        int cts = (int)(cts_f + 0.5); // round to nearest integer
        float err = fabs(cts_f - (float)cts);
        float aval_f = 24000000.0 * (float)n / (128.0 * (float)audio_freq_hz);
        int aval = (int)aval_f; // truncate (round toward zero)
        if ((float)aval == aval_f &&
(err < min_err ||
            ((err == min_err) && (abs(n - ideal_n) < abs((int)best_n - ideal_n)))))
        {
            min_err = err;
            best_n = (NvU32)n;
            best_cts = (NvU32)cts;
            best_a = (NvU32)aval;
        }
    }
```

```
    assert(best_n != 0 && best_cts != 0);
    assert((double)best_cts ==
     ((double)pixclk_freq_hz * best_n) / (128.0 * audio_freq_hz));

}
```

28.7 HDMI Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

28.7.1 HDMI_CTXSW_0

Channel IDs

Context switch register. Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Context switch works like this: Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS (see vmod/chexample). Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be pre-loaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts. Another way to avoid context switch interrupts is to set the AUTO_ACK bit.

This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x0 │ Byte Offset: 0x0 │ Read/Write: R/W │ Reset: 0xXXXXf800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

28.7.2 HDMI_NV_PDISP_SOR_STATE0_0

Writing a 1 to this field causes a 1-cycle pulse, which is used to activate the fields in registers NV_PDISP_SOR_STATE[1,2]. These fields are assembly registers; the active state is located in disp_sor_aa_core.v and disp_or_aaa_super.v

The ATTACHED field in SOR_STATE1 is flopped to create the read-only field "ATTACHED" in the NV_PDISP_SOR_TEST register.

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	UPDATE

28.7.3 HDMI_NV_PDISP_SOR_STATE1_0

Offset: 0x2 | Byte Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
4	0x0	ARM_SHOW_VGA
3	0x0	ATTACHED
2	SAFE	ASY_ORMODE: 0 = SAFE 1 = NORMAL
1:0	0x0	ASY_HEAD_OPMODE: 0 = SLEEP 1 = SNOOZE 2 = AWAKE

28.7.4 HDMI_NV_PDISP_SOR_STATE2_0

Offset: 0x3 | Byte Offset: 0xc | Read/Write: R/W | Reset: 0x00000151 (0bxxxxxxxxxxxxxxxxx000000101010001) | Default: 0x00000030

Bit	Reset	Default	Description
14	0x0	NONE	ASY_DEPOL: 0 = POSITIVE_TRUE 1 = NEGATIVE_TRUE
13	0x0	NONE	ASY_VSYNCPOL: VSYNCPOL depends on the video mode. 0 = POSITIVE_TRUE 1 = NEGATIVE_TRUE
12	0x0	NONE	ASY_HSYNCPOL: HSYNCPOL depends on the video mode. There is no single correct value. There are unlimited modes. 0 = POSITIVE_TRUE 1 = NEGATIVE_TRUE
11:8	SINGLE_TMDS_A	NONE	ASY_PROTOCOL: 1 = SINGLE_TMDS_A 15 = CUSTOM
7:6	COMPLETE_RASTER	NONE	ASY_CRCMODE: 0 = ACTIVE_RASTER 1 = COMPLETE_RASTER 2 = NON_ACTIVE_RASTER
5:4	SUBHEAD0	BOTH	ASY_SUBOWNER: 0 = NONE 1 = SUBHEAD0 2 = SUBHEAD1 3 = BOTH
3:0	HEAD0	NONE	ASY_OWNER: 0 = NONE 1 = HEAD0

28.7.5 HDMI_NV_PDISP_RG_HDCP_AN_MSB_0

HDCP control registers in the pipeline.

HDCP AN MSB REGISTER

The AN_MSB register holds the 32 most significant bits of the link encryption session random number. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0x4 │ Byte Offset: 0x10 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

28.7.6 HDMI_NV_PDISP_RG_HDCP_AN_LSB_0

HDCP AN LSB REGISTER

The AN_LSB register holds the 32 least significant bits of the link encryption session random number. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0x5 │ Byte Offset: 0x14 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

28.7.7 HDMI_NV_PDISP_RG_HDCP_CN_MSB_0

HDCP CN MSB REGISTER

The CN_MSB register holds the 32 most significant bits of the upstream exchange random number. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0x6 │ Byte Offset: 0x18 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE: 0 = ZERO

28.7.8 HDMI_NV_PDISP_RG_HDCP_CN_LSB_0

HDCP CN LSB REGISTER

The CN_LSB register holds the 32 least significant bits of the upstream exchange random number. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0x7 │ Byte Offset: 0x1c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE: 0 = ZERO

28.7.9 HDMI_NV_PDISP_RG_HDCP_AKSV_MSB_0

HDCP AKSV MSB REGISTER

The AKSV_MSB register holds the 8 most significant bits of the transmitter's downstream key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0x8 │ Byte Offset: 0x20 │ Read/Write: RO │ Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: 0 = ZERO

28.7.10 HDMI_NV_PDISP_RG_HDCP_AKSV_LSB_0

HDCP AKSV LSB REGISTER

The AKSV_LSB register holds the 32 least significant bits of the transmitter's downstream key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0x9 │ Byte Offset: 0x24 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

28.7.11 HDMI_NV_PDISP_RG_HDCP_BKSV_MSB_0

HDCP BKSV MSB REGISTER

The BKSV_MSB register holds the 8 most significant bits of the receiver's key selection vector (KSV), as well as the receiver's REPEATER bit. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0xa │ Byte Offset: 0x28 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31	0x0	REPEATER: 0 = VALUE
7:0	0x0	VALUE: 0 = ZERO

28.7.12 HDMI_NV_PDISP_RG_HDCP_BKSV_LSB_0

HDCP BKSV LSB REGISTER

The BKSV_LSB register holds the 32 least significant bits of the receiver's key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0xb │ Byte Offset: 0x2c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE: 0 = ZERO

28.7.13 HDMI_NV_PDISP_RG_HDCP_CKSV_MSB_0

HDCP CKSV MSB REGISTER

The CKSV_MSB register holds the 8 most significant bits of the software's key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Note: Writing CKSV_MSB is the trigger for computing MPRIME and DKSV. It must be written last, after HDCP_CMODE, CN_MSB/LSB, and CKSV_LSB.

Usage: Normal Operation

Offset: 0xc │ Byte Offset: 0x30 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VALUE: 0 = ZERO

28.7.14 HDMI_NV_PDISP_RG_HDCP_CKSV_LSB_0

HDCP CKSV LSB REGISTER

The CKSV_LSB register holds the 32 least significant bits of the software's key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0xd │ Byte Offset: 0x34 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VALUE: 0 = ZERO

28.7.15 HDMI_NV_PDISP_RG_HDCP_DKSV_MSB_0

HDCP DKSV MSB REGISTER

The DKSV_MSB register holds the 8 most significant bits of the transmitter's upstream key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0xe │ Byte Offset: 0x38 │ Read/Write: RO │ Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VALUE: 0 = ZERO

28.7.16 HDMI_NV_PDISP_RG_HDCP_DKSV_LSB_0

HDCP DKSV LSB REGISTER

The DKSV_LSB register holds the 32 least significant bits of the transmitter's upstream key selection vector (KSV). See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0xf │ Byte Offset: 0x3c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

28.7.17 HDMI_NV_PDISP_RG_HDCP_CTRL_0

HDCP CTRL REGISTER

The CTRL register is used to facilitate synchronization between the display driver and hardware.

- RUN: Set by the driver to start or stop the downstream protocol
- CRYPT: Set by the driver to turn encryption on.
- Write ENABLED to enable, but only after RUN == YES. Once enabled, it is disabled by writing RUN=NO.
- DUAL_LINK_EN: Set by the driver to turn dual-link mode on or off
- ONEONE: ONEONE_EN enables the HDCP 1.1 features. This enables HDCP EESS signaling and Advance_Cipher.

- Write ENABLED to enable, but only after RUN == YES. Once enabled, it is disabled by writing RUN=NO. This should be enabled if your receiver supports HDCP 1.1

See Chapter 2.7 in the HDCP 1.1 specification for EESS signaling. See Chapter 2.2 in the HDCP 1.1 specification for ADVANCE_CIPHER.

- AN: Set by hardware when a valid An value has been generated
- R0: Set by hardware when Km, Ks, M0, and R0 have been calculated
- SPRIME: Set by hardware when S' has been calculated
- MPRIME: Set by hardware when M' has been calculated
- SROM_EN: Set by hardware while the HDCP SROM is in use, cleared when not in use
- SROM_ERR: Set by hardware when an error occurs while using the HDCP SROM

Usage: Normal operation

Offset: 0x10 │ Byte Offset: 0x40 │ Read/Write: R/W │ Reset: 0x0000XX00 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	R/W	Reset	Description
13	RO	X	SROM_ERR: 0 = NOERROR 1 = ERROR
12	RO	X	SROM_EN: 0 = DISABLED 1 = ENABLED
11	RO	X	MPRIME: 0 = INVALID 1 = VALID
10	RO	X	SPRIME: 0 = INVALID 1 = VALID
9	RO	X	R0: 0 = INVALID 1 = VALID
8	RO	X	AN: 0 = INVALID 1 = VALID
3	RW	0x0	ONEONE: 0 = DISABLED 1 = ENABLED
2	RW	0x0	DUAL_LINK_EN: 0 = DISABLED 1 = ENABLED
1	RW	0x0	CRYPT: 0 = DISABLED 1 = ENABLED
0	RW	0x0	RUN: 0 = NO 1 = YES

28.7.18 HDMI_NV_PDISP_RG_HDCP_CMODE_0

HDCP CMODE REGISTER

The CMODE register indicates to hardware which upstream protocol calculation to perform. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Writing to the CMODE register kicks off a "Read Status" or a "Read M" operation. When performing a "Read Status" operation, use the CMODE_INDEX field to identify the port for which you wish to obtain status.

Usage: Normal Operation

Offset: 0x11 │ Byte Offset: 0x44 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000001)

Bit	Reset	Description
7:4	0x0	INDEX: 0 = TMDS0_LINK1 1 = TMDS0_LINK0 2 = TMDS1_LINK1 3 = TMDS1_LINK0 4 = DVOA 5 = DVOB 6 = DAC1 7 = DAC2 8 = DAC3 9 = TMDS2_LINK1 10 = TMDS2_LINK0 11 = DP2_LINK1 12 = DP2_LINK0 13 = DP1_LINK1 14 = DP1_LINK0
3:0	0x1	MODE: 1 = READ_S 2 = READ_M

28.7.19 HDMI_NV_PDISP_RG_HDCP_MPRIME_MSB_0

HDCP MPRIME MSB REGISTER

The MPRIME_MSB register holds the 32 most significant bits of the encrypted M0 value. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0x12 │ Byte Offset: 0x48 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

28.7.20 HDMI_NV_PDISP_RG_HDCP_MPRIME_LSB_0

HDCP MPRIME LSB REGISTER

The MPRIME_LSB register holds the 32 least significant bits of the encrypted M0 value. See the HDCP specification and the Upstream Link for HDCP specification document for more information.

Usage: Normal Operation

Offset: 0x13 │ Byte Offset: 0x4c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

28.7.21 HDMI_NV_PDISP_RG_HDCP_SPRIME_MSB_0

HDCP SPRIME MSB REGISTER

The SPRIME_MSB register holds the 8 most significant bits of the S' value. See the HDCP specification and the Upstream Link for HDCP specification for more information.

STATUS_READZ indicates whether or not this chip implements the HDCP Upstream Spec's "Read Z" operation. Currently, "Read Z" is not implemented.

STATUS_CS indicates whether or not this chip implements the connection state (CS) register. If this field is set to STATUS_CS_NOT_IMPL, it means this chip will always follow the standard "Read Status" protocol when reporting status. If this field is set to STATUS_CS_IMPLMNTD, it means this chip will always follow the "Read Status With Connection State" protocol when reporting status. STATUS_SCOPE indicates how the chip will report the status for the STATUS_UNPROTECTED field. See the STATUS_UNPROTECTED description below for more information.

STATUS_INTPNL indicates if the port you are querying (identified by the CMode index) is transmitting to an internal panel on this head. If the port you are querying is not transmitting on this head (see the ** note below), or if it is transmitting on this head but to a device other than an internal panel, then STATUS_INTPNL_INACTV will be returned.

STATUS_MAX_CMODE_IDX identifies the maximum CMode index allowed for requesting status.

> ****Note:** Unlike the STATUS_UNPROTECTED field, the other STATUS fields always use the _HA_ registers to report status for ports on head A and the _HB_ registers to report status on head B. If you use the _HA_ versions of the CMODE and SPRIME registers to query the status of a port that is actually being driven by head B, the status will indicate that the port isn't transmitting at all (everything will come back as INACTV).

Usage: Normal operation

Offset: 0x14 │ Byte Offset: 0x50 │ Read/Write: RO │ Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7	X	STATUS_READZ: 0 = NOT_IMPL 1 = IMPLMNTD
6	X	STATUS_CS: 0 = NOT_IMPL 1 = IMPLMNTD
5	X	STATUS_SCOPE: 0 = SCOPE_TWO_HEADS 1 = SCOPE_1_HEAD

Bit	Reset	Description
4	X	STATUS_INTPNL: 0 = INACTV 1 = ACTV
3:0	X	STATUS_MAX_CMODE_IDX

28.7.22 HDMI_NV_PDISP_RG_HDCP_SPRIME_LSB2_0

HDCP SPRIME LSB2 REGISTER

The SPRIME_LSB2 register holds the 32 next-most significant bits of the S' value. See the HDCP specification and the Upstream Link for HDCP specification for more information.

STATUS_CMODE_IDX identifies the index of the port to which the status request was actually routed. STATUS_UNPROTECTED indicates whether the port you queried (identified by the CMode index) is transmitting unprotected data. If STATUS_SCOPE is set to STATUS_SCOPE_1_HEAD, then make sure to use the _HA_ registers to see if any ports on head A are unprotected, and use the _HB_ registers to see if any ports on head B are unprotected. If STATUS_SCOPE is set to STATUS_SCOPE_2_HEADS, then STATUS_UNPROTECTED combines the status of both heads. In this case, both the _HA_ registers and the _HB_ registers will report the same value for the STATUS_UNPROTECTED field.

Value of STATUS_SCOPE	Meaning of STATUS_UNPROTECTED
_1_HEAD	UNPROTECTED_Y means that at least one port driven by this head is transmitting "unprotected" data. UNPROTECTED_N means that none of the ports driven by this head are transmitting "unprotected" data.
_2_HEADS	UNPROTECTED_Y means that at least one port driven by either head is transmitting "unprotected" data. UNPROTECTED_N means that none of the ports driven by either head are transmitting "unprotected" data.

"Unprotected" data does not necessarily mean data that is not being encrypted. Data being transmitted out of a DAC is *always* considered unprotected. Data being transmitted out of a DVO is considered unprotected if it is not being encrypted. Data being transmitted by a TMDS link is also considered unprotected if it is not being encrypted,*except* for the case where the link feeds the internal panel of a laptop. The HDCP specification dictates that since the bus feeding an internal panel is essentially inaccessible to users, it can be considered "protected" even when sending clear data.

STATUS_EXTPNL is set to STATUS_EXTPNL_ACTV if the port identified by the CMode index is a digital interface (i.e., a DVO or TMDS interface), and it is transmitting on this head to something *other* than an internal panel. The STATUS_INTPNL field and the STATUS EXTPNL field will never be both set to ACTV for the same port. If a digital port is transmitting on this head, and it is physically connected to an internal panel, then the STATUS_INTPNL bit is set to ACTV and the STATUS_EXTPNL is set to INACTV; if it is transmitting on this head but it is *not* connected to an internal panel (even if it is not connected to anything at all), then STATUS_INTPNL is set to INACTV and STATUS_EXTPNL is set to ACTV.

STATUS_RPTR is set to STATUS_RPTR_ACTV if 1) the STATUS_EXTPNL field is set to ACTV, and 2) the REPEATER bit of the BKSV_MSB register is set to 0x1. This scenario should only occur after we are transmitting to an external HDCP-compliant device whose "repeater" bit is set.

STATUS_ENCRYPTING will be set to STATUS_ENCRYPTING_Y if the HDCP unit in this head is actually encrypting the data it receives. Note that if this bit is set, it means that *all* ports driven by this head whose data streams flow through the HDCP unit will be transmitting encrypted data, but any ports driven by this head whose data streams *bypass* the HDCP unit will be transmitting *clear* data. The DACs always use a data path that bypasses the HDCP unit and therefore must always be considered "unprotected".

Usage: Normal Operation

Offset: 0x15 | Byte Offset: 0x54 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	STATUS_CMODE_IDX
27	X	STATUS_UNPROTECTED: 0 = N 1 = Y
26	X	STATUS_EXTPNL: 0 = INACTV 1 = ACTV
25	X	STATUS_RPTR: 0 = INACTV 1 = ACTV
24	X	STATUS_ENCRYPTING: 0 = N 1 = Y
23:0	X	VALUE: 0 = ZERO

28.7.23 HDMI_NV_PDISP_RG_HDCP_SPRIME_LSB1_0

HDCP SPRIME LSB1 REGISTER

The SPRIME_LSB1 register holds the 32 least significant bits of the S' value. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0x16 | Byte Offset: 0x58 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

28.7.24 HDMI_NV_PDISP_RG_HDCP_RI_0

HDCP RI REGISTER

The RI register holds the 16-bit link integrity check value. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0x17 | Byte Offset: 0x5c | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	VALUE: 0 = ZERO

28.7.25 HDMI_NV_PDISP_RG_HDCP_CS_MSB_0

HDCP CS MSB REGISTER

The CS_MSB register holds the 8 most significant bits of the connection state. See the HDCP specification and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0x18 │ Byte Offset: 0x60 │ Read/Write: RO │ Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	Reserved

28.7.26 HDMI_NV_PDISP_RG_HDCP_CS_LSB_0

HDCP CS LSB REGISTER

The CS_LSB register holds the 32 least significant bits of the connection state. See the HDCP specification, and the Upstream Link for HDCP specification for more information.

Usage: Normal Operation

Offset: 0x19 │ Byte Offset: 0x64 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RESERVED

28.7.27 HDMI_NV_PDISP_HDMI_AUDIO_EMU0_0

HDMI_AUDIO_EMU

HDMI_AUDIO_EMU0 and HDMI_AUDIO_EMU1 are used to program the S/PDIF transmitter during RTL simulation and emulation. This is strictly for verification. This register is write-only.

Offset: 0x1a │ Byte Offset: 0x68 │ Read/Write: WO │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	REGX

28.7.28 HDMI_NV_PDISP_HDMI_AUDIO_EMU_RDATA0_0

HDMI_AUDIO_EMU_RDATA0 is read-only.

Offset: 0x1b │ Byte Offset: 0x6c │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RDATA

28.7.29 HDMI_NV_PDISP_HDMI_AUDIO_EMU1_0

HDMI_AUDIO_EMU1 and EMU2 allow control over S/PDIF transmitter clock ratios (which are fixed in the GPU). They do indirect addressing.

EMU1 has the address, and EMU2 has the data.

To write:

1. Write data register with data

2. Write address register with bit 31=1, and address in 15:0

3. Write address register with bit 31=0

To read:

1. Write address register with bit 31=0 and address in 15:0

2. Read back address register to introduce delay allowing read data to propagate

3. Read data register

Offset: 0x1c │ Byte Offset: 0x70 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
31	0x0	WRITE
7:0	0x0	ADDR: 0 = CNT_22 1 = CNT_24 2 = CNT_32 3 = CNT_44 4 = CNT_48 5 = CNT_88 6 = CNT_96 7 = CNT_176 8 = CNT_192 9 = JIT_05_22 10 = JIT_05_24 11 = JIT_05_32 12 = JIT_05_44 13 = JIT_05_48 14 = JIT_05_88 15 = JIT_05_96 16 = JIT_05_176 17 = JIT_05_192 18 = JIT_10_22 19 = JIT_10_24 20 = JIT_10_32 21 = JIT_10_44 22 = JIT_10_48 23 = JIT_10_88 24 = JIT_10_96 25 = JIT_10_176 26 = JIT_10_192 27 = JIT_11_22 28 = JIT_11_24 29 = JIT_11_32 30 = JIT_11_44 31 = JIT_11_48 32 = JIT_11_88 33 = JIT_11_96 34 = JIT_11_176 35 = JIT_11_192 36 = GLITCH_PERIOD_22 37 = GLITCH_PERIOD_24 38 = GLITCH_PERIOD_32 39 = GLITCH_PERIOD_44 40 = GLITCH_PERIOD_48

Bit	Reset	Description
		41 = GLITCH_PERIOD_88 42 = GLITCH_PERIOD_96 43 = GLITCH_PERIOD_176 44 = GLITCH_PERIOD_192 45 = GLITCH_CNT_22 46 = GLITCH_CNT_24 47 = GLITCH_CNT_32 48 = GLITCH_CNT_44 49 = GLITCH_CNT_48 50 = GLITCH_CNT_88 51 = GLITCH_CNT_96 52 = GLITCH_CNT_176 53 = GLITCH_CNT_192

28.7.30 HDMI_NV_PDISP_HDMI_AUDIO_EMU2_0

Offset: 0x1d │ Byte Offset: 0x74 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

28.7.31 HDMI_NV_PDISP_HDMI_AUDIO_INFOFRAME_CTRL_0

This is the register space for the HDMI block. Refer to HDMI Version 1.1. Refer to IEC60958-1 for general information and IEC60958-3 for consumer specifications for related audio information.

Additional notes regarding the ACR packet:

There are several methods that can be used to generate the N and CTS values in the Audio Clock Regeneration packet. The details of these methods are described here.

Full Software control: Software controls the contents of the possible ACR packets. Software writes the contents of the ACR_XXXX_SUBPACK_HIGH and ACR_XXXX_SUBPACK_LOW for all seven audio frequencies.

Software selects a method in ACR_CTRL to determine the audio sampling frequency

1. Measured in the audio block (MEASURE)
2. Read from the channel status information encoded in the audio stream (PACKET)
3. Defined by software (FREQS)

Software writes _YES to the ENABLE field of the SUBPACK_HIGH registers to allow that pair of registers to be used as the ACR packet. The audio sampling frequency specified in ACR_CTRL will select one of the seven pairs of SUBPACK registers to use as the ACR packet. This packet is sent on every Frame 27 of the audio block if that pair's ENABLE field is _YES.

Hardware controlled CTS: Software provides the N value and the hardware measures the CTS value. This is described in the HDMI_SPARE comments and mentioned in all other affected registers.

Software writes _ENABLE to HDMI_SPARE_HW_CTS to enable this feature.

Software writes 1 to HDMI_SPARE_CTS_RESET_VAL. This controls what the internal CTS counter resets to when it starts the next round of measurement. Resetting to zero was not correct, and adding a few bits of control helped to fine tune the measurement.

Software sets all ENABLEs in ACR_CTRL to _NO. (The easy way to do this is just to write zero to the whole register.)

Software writes the value of N in two places:

1. ACR_0441_SUBPACK_HIGH_SB[4/5/6]: It is written here so that the ACR packet contains the correct information for N.

2. AUDIO_N_VALUE: It is written here so that the audio block knows what N to use while it is measuring CTS. (Software does not need to write N to these places if Hardware Selected N is used.)

Software enables the sending of ACR_0441_SUBPACK* by setting ACR_0441_SUBPACK_ENABLE to _YES. (This pair of registers is used for all audio frequencies when Hardware controlled CTS is enabled.)

Software writes _USE_HW_CTS_VAL to ACR_0441_SUBPACK_LOW_SB1.

Hardware selected N: Software provides the possible N values whenever the pixel clock frequency changes. Hardware looks up the correct N value from a table stored in priv registers.

Software writes the 7 possible N values into the 7 AUDIO_NVAL registers. This is based on pixel clock frequency.

Software enables this feature by setting AUDIO_N_LOOKUP to _ENABLE. Hardware will use the audio sampling frequency detected by the audio block to select the correct N value.

NOTE: This is not the audio sampling frequency reported by the channel status information in the audio stream. Hardware will ignore whatever if written in the SUBPACK_HIGH registers when this feature is enabled and send the value of N it selected in these bytes of the ACR packet. CTS can come from the SUBPACK_LOW registers or the Hardware controlled CTS.

Audio frequency notes: There are several places in the HDMI and Audio registers where the audio sampling frequency is reported or set. Here is a breakdown of all such fields.

In SPDIF audio encoding, one bit of each audio sample is part of the "channel status" information. The bits from all 192 audio samples combine to form the 192 bit channel status packet. For consumer applications, only the first 40 bits have defined values. These bits are included in the encoded HDMI audio packet. Four of these bits contain the audio sampling frequency as reported by the audio source. See the IEC60958-3 specification for more information.

SPDIF_CHN_STATUS1 and SPDIF_CHN_STATUS2 are read-only registers that contain the 40 bits of channel status data read from the last audio block. The audio frequency the audio stream reports can be read from SPDIF_CHN_STATUS1_SFREQ. This is the sampling frequency that ACR_CTRL refers to with _PACKET.

SPDIF_CHN_STATUS1_ORIGINAL is the original sampling frequency of the audio. If the source of the audio stream changed the sampling frequency from its original frequency for any reason, the original sampling frequency is reported here.

AUDIO_CNTRL0_SAMPLING_FREQ is the sampling frequency measured by the audio block. It counts the number of dispclks periods that occur between to specific points in the audio stream. It compares this count to the thresholds in the AUDIO_FS registers to determine what the audio sampling frequency is. This is the sampling frequency that ACR_CTRL refers to as _MEASURE. This is the sampling frequency used to determine the correct value of N when Hardware selected N is enabled (see AUDIO_N).

The CHANNEL_STATUS1 and CHANNEL_STATUS2 registers are for debug. They can be used to replace the channel status bits that were in the original SPDIF stream with user-defined bits. CHANNEL_STATUS1_SFREQ can be used to report a different sampling frequency to downstream devices.

ACR_CTRL_FREQS can be written to select a specific pair of SUBPACK registers to send as your ACR packet. ACR_CTRL_FREQS_ENABLE must be set to _YES for this to have any effect.

The HDMI Audio InfoFrame, AVI InfoFrame, and Generic InfoFrame have nearly identical priv register interfaces. The next five registers control the Audio InfoFrame as described in section 8.2.2 of the HDMI specification.

HDMI_AUDIO_INFOFRAME_CTRL

This register controls the frequency and generation of audio InfoFrame packets. The contents of the packet should be written into the header (HDMI_AUDIO_INFOFRAME_HEADER) and subpacket (HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW, HDMI_AUDIO_INFOFRAME_SUBPACK0_HIGH) registers.

ENABLE: Setting this field to _YES will initiate InfoFrame generation. Setting this bit to _NO will disable InfoFrame generation at the beginning of the next frame. The frequency of InfoFrame generation is controlled by OTHER and SINGLE fields.

OTHER: Setting this field to _EN while SINGLE is set to _DIS will cause InfoFrame to be transmitted to every other frame.

SINGLE: Setting this field to _EN while OTHER is set to _DIS will cause InfoFrame to be transmitted exactly once.

If OTHER and SINGLE fields are both set to _DIS, an InfoFrame will be generated every frame. Software should never set OTHER and SINGLE both to _EN.

See chapter 8.2.2 - Audio InfoFrame, in the HDMI specification for more information

CHKSUM_HW: Hardware provides a way to calculate the checksum for the InfoFrames.

_ENABLE will enable the hardware calculation to be passed to the packet

_DISABLE:

- Uses the register value defined in NV_PDISP_SOR_HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW_PB0 for the S/PDIF
- Uses the checksum provided by the Azalia codec for the Azalia audio formats.

Usage: Normal Operation

Offset: 0x1e │ Byte Offset: 0x78 │ Read/Write: R/W │ Reset: 0x00000200 (0bxxxxxxxxxxxxxxxxxxxxxxxx10xxx0xxx0)

Bit	Reset	Description
9	ENABLE	CHKSUM_HW: CHKSUM_HW is an optional feature for software to use. Not required to be enabled. 0 = DISABLE 1 = ENABLE
8	0x0	SINGLE: 0 = DIS 1 = EN
4	0x0	OTHER: 0 = DIS 1 = EN
0	0x0	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

28.7.32 HDMI_NV_PDISP_HDMI_AUDIO_INFOFRAME_STATUS_0

HDMI_AUDIO_INFOFRAME_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in INFOFRAME_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

Usage: Normal Operation

Offset: 0x1f │ Byte Offset: 0x7c │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

28.7.33 HDMI_NV_PDISP_HDMI_AUDIO_INFOFRAME_HEADER_0

HDMI_AUDIO_INFOFRAME_HEADER

This register should be written with the value of the HDMI Audio InfoFrame header. This header is described in table 8-4 of chapter 8.2.2 of the HDMI specification. HB0, HB1, and HB2 are defined fields of the header from the specification.

Offset: 0x20 │ Byte Offset: 0x80 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

28.7.34 HDMI_NV_PDISP_HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW_0

HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW

This register should be written with the lower 4 bytes of the HDMI Audio InfoFrame. PB0, PB1, PB2, and PB3 are defined fields from the specification. See chapter 8.2.2 - Audio InfoFrame, Table 8-5, in the HDMI specification for more information

Offset: 0x21 │ Byte Offset: 0x84 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

28.7.35 HDMI_NV_PDISP_HDMI_AUDIO_INFOFRAME_SUBPACK0_HIGH_0

HDMI_AUDIO_INFOFRAME_SUBPACK0_HIGH

This register should be written with the upper 2 bytes of the HDMI Audio InfoFrame. PB4 and PB5 are defined fields from the specification. See chapter 8.2.2 - Audio InfoFrame, Table 8-5, in the HDMI specification for more information

Offset: 0x22 │ Byte Offset: 0x88 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	PB5
7:0	0x0	PB4

28.7.36 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_CTRL_0

The following registers are used to send a 14-byte packet intended for sending AVI InfoFrame packet as described in section 8.2.1 of the HDMI specification.

HDMI_AVI_INFOFRAME_CTRL

This register controls the frequency and generation of AVI InfoFrame packets. The fields of this register are identical to HDMI_AUDIO_INFOFRAME_CTRL. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, in the HDMI specification for more information

Offset: 0x23 | Byte Offset: 0x8c | Read/Write: R/W | Reset: 0x00000200 (0bxxxxxxxxxxxxxxxxxxxxxxxx10xxx0xxx0)

Bit	Reset	Description
9	ENABLE	CHKSUM_HW: CHKSUM_HW is an optional feature for software to use. Not required to be enabled 0 = DISABLE 1 = ENABLE
8	DIS	SINGLE: 0 = DIS 1 = EN
4	DIS	OTHER: 0 = DIS 1 = EN
0	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

28.7.37 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_STATUS_0

HDMI_AVI_INFOFRAME_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in INFOFRAME_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

Offset: 0x24 | Byte Offset: 0x90 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

28.7.38 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_HEADER_0

HDMI_AVI_INFOFRAME_HEADER

This register should be written with the value of the HDMI AVI InfoFrame header. This header is described in table 8-1 of chapter 8.2.1 of the HDMI specification. HB0, HB1, and HB2 are defined fields of the header from the specification.

Offset: 0x25 | Byte Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

28.7.39 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_SUBPACK0_LOW_0

HDMI_AVI_INFOFRAME_SUBPACK0_LOW

This register should be written with the lower 4 bytes of the HDMI AVI InfoFrame. PB0, PB1, PB2, and PB3 are defined fields from the specification. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI specification for more information.

Offset: 0x26 | Byte Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

28.7.40 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_SUBPACK0_HIGH_0

HDMI_AVI_INFOFRAME_SUBPACK0_HIGH

This register should be written with bytes 4-6 of the HDMI AVI InfoFrame. PB3, PB5, and PB6 are defined fields from the specification. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI specification for more information.

Offset: 0x27 | Byte Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB6
15:8	0x0	PB5
7:0	0x0	PB4

28.7.41 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_SUBPACK1_LOW_0

HDMI_AVI_INFOFRAME_SUBPACK1_LOW

This register should be written with bytes 7-10 of the HDMI AVI InfoFrame. PB7, PB8, PB9, and PB10 are defined fields from the specification. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI specification for more information.

Offset: 0x28 | Byte Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB10
23:16	0x0	PB9
15:8	0x0	PB8
7:0	0x0	PB7

28.7.42 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_SUBPACK1_HIGH_0

HDMI_AVI_INFOFRAME_SUBPACK1_HIGH

This register should be written with bytes 11-13 of the HDMI AVI InfoFrame. PB11, PB12, and PB13 are defined fields from the specification. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI specification for more information.

Offset: 0x29 │ Byte Offset: 0xa4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB13
15:8	0x0	PB12
7:0	0x0	PB11

28.7.43 HDMI_NV_PDISP_HDMI_GENERIC_CTRL_0

The following registers are used to send a 28-byte packet intended for sending any packet type. These registers will most likely be used for debug purposes.

HDMI_GENERIC_CTRL

This register controls the frequency and generation of generic InfoFrame packets. The fields of this register are identical to HDMI_AUDIO_INFOFRAME_CTRL except where noted below.

HBLANK: If HBLANK is set to _EN, then this packet will be sent (once) during the next horizontal blanking interval. The packet will still be sent at most once per frame. Software can poll GENERIC_STATUS_SENT to determine when the packet has been sent.

Using HBLANK is intended to mimic Audio Sample Packets and ACR Packets and therefore sending of the actual audio packets should be disabled by setting AUDIO to _DIS. This feature is used for debug purposes only.

To mimic an audio packet:

- begin
- wait for GENERIC_STATUS_SENT=_WAITING
- write next 4 audio samples to GENERIC_HEADER/GENERIC_SUBPACK registers
- set HBLANK = _EN
- wait for GENERIC_STATUS_SENT=_SENT
- set HBLANK = _DIS
- end

AUDIO: Audio packet transmission will be stopped when set to _DIS. Normal audio packet transmission will be allowed when set to _EN. This is used during debug to disable normal audio when using HBLANK.

Offset: 0x2a │ Byte Offset: 0xa8 │ Read/Write: R/W │ Reset: 0x00010000 (0bxxxxxxxxxxxxxxxx1xxx0xxx0xxx0xxx0)

Bit	Reset	Description
16	EN	AUDIO: Set to 1 in audio mode and 0 in DVI mode. 0 = DIS 1 = EN
12	DIS	HBLANK: 0 = DIS 1 = EN
8	DIS	SINGLE: 0 = DIS 1 = EN
4	DIS	OTHER: 0 = DIS 1 = EN

Bit	Reset	Description
0	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

28.7.44 HDMI_NV_PDISP_HDMI_GENERIC_STATUS_0

HDMI_GENERIC_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in INFOFRAME_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

Offset: 0x2b │ Byte Offset: 0xac │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

28.7.45 HDMI_NV_PDISP_HDMI_GENERIC_HEADER_0

HDMI_GENERIC_HEADER

This register should be written with the contents of the packet header.

Offset: 0x2c │ Byte Offset: 0xb0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

28.7.46 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK0_LOW_0

HDMI_GENERIC_SUBPACK0_LOW

Bytes 0-3 of the packet are written into this register.

Offset: 0x2d │ Byte Offset: 0xb4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

28.7.47 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK0_HIGH_0

HDMI_GENERIC_SUBPACK0_HIGH

Bytes 4-6 of the packet are written into this register.

Offset: 0x2e │ Byte Offset: 0xb8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB6
15:8	0x0	PB5
7:0	0x0	PB4

28.7.48 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK1_LOW_0

HDMI_GENERIC_SUBPACK1_LOW

Bytes 7-10 of the packet are written into this register.

Offset: 0x2f │ Byte Offset: 0xbc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB10
23:16	0x0	PB9
15:8	0x0	PB8
7:0	0x0	PB7

28.7.49 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK1_HIGH_0

HDMI_GENERIC_SUBPACK1_HIGH

Bytes 11-13 of the packet are written into this register.

Offset: 0x30 │ Byte Offset: 0xc0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB13
15:8	0x0	PB12
7:0	0x0	PB11

28.7.50 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK2_LOW_0

HDMI_GENERIC_SUBPACK2_LOW

Bytes 14-17 of the packet are written into this register.

Offset: 0x31 │ Byte Offset: 0xc4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB17
23:16	0x0	PB16
15:8	0x0	PB15

Bit	Reset	Description
7:0	0x0	PB14

28.7.51 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK2_HIGH_0

HDMI_GENERIC_SUBPACK2_HIGH

Bytes 18-20 of the packet are written into this register.

Offset: 0x32 | Byte Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB20
15:8	0x0	PB19
7:0	0x0	PB18

28.7.52 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK3_LOW_0

HDMI_GENERIC_SUBPACK3_LOW

Bytes 21-24 of the packet are written into this register.

Offset: 0x33 | Byte Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	PB24
23:16	0x0	PB23
15:8	0x0	PB22
7:0	0x0	PB21

28.7.53 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK3_HIGH_0

HDMI_GENERIC_SUBPACK3_HIGH

Bytes 25-27 of the packet are written into this register.

Offset: 0x34 | Byte Offset: 0xd0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000000000000000000)

Bit	Reset	Description
23:16	0x0	PB27
15:8	0x0	PB26
7:0	0x0	PB25

28.7.54 HDMI_NV_PDISP_HDMI_ACR_CTRL_0

HDMI_ACR_CTRL

The Audio Clock Regeneration (ACR) packet contains the N and CTS values that the HDMI sink requires to recreate the audio clock. This mechanism is described in chapter 7.2 of the HDMI specification.

If software needs to specify N and CTS directly, this register is used to select the audio sampling frequency detection mechanism. The sampling rate is used to index one of the seven ACR_XXXX_SUBPACK_LOW/HIGH registers which must be

written with the correct value of N and CTS. N and CTS are determined by the current audio sampling frequency and the pixel clock frequency:

CTS = (PixelClock*N) / (128*AudioSamplingFrequency)

The selected sampling rate must point to a valid entry (i.e., one of the 7 listed below) and the selected ACR_XXXX_SUBPACK_HIGH_ENABLE must be set to enable sending the packet.

When software is controlling N and CTS directly, the ACR packet is sent every 27th audio frame (of 0 to 191) of the audio block. The sampling frequency read from the channel status bits is not available until the 27th audio frame.

PACKET_ENABLE: Set this to _YES to use the channel status information read from the incoming SPDIF audio stream to determine the sampling frequency.

MEASURE_ENABLE: Set this to _YES to use the sampling frequency measured in the in the audio block to determine the sampling frequency. This is the sampling frequency that is read from AUDIO_CNTRL0_SAMPLING_FREQ.

FREQS_ENABLE: Set this to _YES to use the sampling frequency written into the _FREQS field of this register.

FREQS: This is the audio sampling frequency that will be used when FREQS_ENABLE is _YES.

When Hardware is used to measure CTS, all ENABLE fields of this register should be set to _NO.

All four subpackets contain the same ACR packet.

Offset: 0x35 │ Byte Offset: 0xd4 │ Read/Write: R/W │ Reset: 0x02010000 (0bxxxx0010xxxxxxx1xxxxxxx0xxxxxxx0)

Bit	Reset	Description
27:24	FREQ_48KHZ	FREQS: 3 = FREQ_32KHZ 0 = FREQ_44_1KHZ 2 = FREQ_48KHZ 8 = FREQ_88_2KHZ 10 = FREQ_96KHZ 12 = FREQ_176_4KHZ 14 = FREQ_192KHZ
16	YES	FREQS_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
8	NO	MEASURE_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
0	NO	PACKET_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

28.7.55 HDMI_NV_PDISP_HDMI_ACR_0320_SUBPACK_LOW_0

HDMI_ACR_0320_SUBPACK_LOW

Contains bytes 1-3 of the 32 kHz ACR packet. This is the CTS value.

See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0x36 │ Byte Offset: 0xd8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

28.7.56 HDMI_NV_PDISP_HDMI_ACR_0320_SUBPACK_HIGH_0

HDMI_ACR_0320_SUBPACK_HIGH

Contains bytes 4-6 of the 32 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x37 │ Byte Offset: 0xdc │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

28.7.57 HDMI_NV_PDISP_HDMI_ACR_0441_SUBPACK_LOW_0

HDMI_ACR_0441_SUBPACK_LOW

Contains bytes 1-3 of the 44.1 kHz ACR packet. This is the CTS value. If Hardware measured CTS is enabled, SB1 should be set to zero. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0x38 │ Byte Offset: 0xe0 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1: 0 = USE_HW_CTS_VAL
23:16	0x0	SB2
15:8	0x0	SB3

28.7.58 HDMI_NV_PDISP_HDMI_ACR_0441_SUBPACK_HIGH_0

HDMI_ACR_0441_SUBPACK_HIGH

Contains bytes 4-6 of the 44.1 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

If Hardware measured CTS is enabled, ACR_0441_SUBPACK_HIGH_N should be written with the N value for the current audio sampling frequency.

If the hardware N value selection is enabled, the N value does not need to be written to this register.

ENABLE: _YES allows this packet to be sent. This should be set to _YES when hardware measured CTS is being used. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x39 │ Byte Offset: 0xe4 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

28.7.59 HDMI_NV_PDISP_HDMI_ACR_0882_SUBPACK_LOW_0

HDMI_ACR_0882_SUBPACK_LOW

Contains bytes 1-3 of the 88.2 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0x3a │ Byte Offset: 0xe8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

28.7.60 HDMI_NV_PDISP_HDMI_ACR_0882_SUBPACK_HIGH_0

HDMI_ACR_0882_SUBPACK_HIGH

Contains bytes 4-6 of the 88.2 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x3b │ Byte Offset: 0xec │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

28.7.61 HDMI_NV_PDISP_HDMI_ACR_1764_SUBPACK_LOW_0

HDMI_ACR_1764_SUBPACK_LOW

Contains bytes 1-3 of the 176.4 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0x3c │ Byte Offset: 0xf0 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000xxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

28.7.62 HDMI_NV_PDISP_HDMI_ACR_1764_SUBPACK_HIGH_0

HDMI_ACR_1764_SUBPACK_HIGH

Contains bytes 4-6 of the 176.4 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x3d │ Byte Offset: 0xf4 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

28.7.63 HDMI_NV_PDISP_HDMI_ACR_0480_SUBPACK_LOW_0

HDMI_ACR_0480_SUBPACK_LOW

Contains bytes 1-3 of the 48 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0x3e │ Byte Offset: 0xf8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

28.7.64 HDMI_NV_PDISP_HDMI_ACR_0480_SUBPACK_HIGH_0

HDMI_ACR_0480_SUBPACK_HIGH

Contains bytes 4-6 of the 48 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x3f │ Byte Offset: 0xfc │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

28.7.65 HDMI_NV_PDISP_HDMI_ACR_0960_SUBPACK_LOW_0

HDMI_ACR_0960_SUBPACK_LOW

Contains bytes 1-3 of the 96 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 of the HDMI specification for more information.

Offset: 0x40 │ Byte Offset: 0x100 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

28.7.66 HDMI_NV_PDISP_HDMI_ACR_0960_SUBPACK_HIGH_0

HDMI_ACR_0960_SUBPACK_HIGH

Contains bytes 4-6 of the 96 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x41 │ Byte Offset: 0x104 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

28.7.67 HDMI_NV_PDISP_HDMI_ACR_1920_SUBPACK_LOW_0

HDMI_ACR_1920_SUBPACK_LOW

Contains bytes 1-3 of the 192 kHz ACR packet. This is the CTS value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

Offset: 0x42 │ Byte Offset: 0x108 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000xxxxxxxx)

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

28.7.68 HDMI_NV_PDISP_HDMI_ACR_1920_SUBPACK_HIGH_0

HDMI_ACR_1920_SUBPACK_HIGH

Contains bytes 4-6 of the 192 kHz ACR packet. This is the N value. See table 5-11 in chapter 5.3.3 in the HDMI specification for more information.

ENABLE: _YES allows this packet to be sent. When set to _NO, this ACR packet will not be sent even if the sampling frequency defined in ACR_CTRL points to this register set.

Offset: 0x43 │ Byte Offset: 0x10c │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

Bit	Reset	Description
23:16	0x0	SB4
15:8	0x0	SB5
7:0	0x0	SB6

28.7.69 HDMI_NV_PDISP_HDMI_CTRL_0

HDMI_CTRL

REKEY: REKEY is the number of clocks required for HDCP rekey, starting from when DE is deasserted. No HDMI packets can be sent during this time. Due to a two cycle delay in hardware, REKEY should be set to two less than the desired value.

AUDIO_LAYOUT: AUDIO_LAYOUT controls layout (HB1[4]) of the Audio Sample Packet Header. Two different layouts are supported in the HDMI specification. The hardware only supports layout _2CH (2 channel audio). This should not need to be programmed beyond its init value of _2CH.

See Section 5.3.4 of the HDMI specification.

AUDIO_LAYOUT_SELECT: For projects with integrated audio codec, the AUDIO_LAYOUT information is automatically detected by hardware (default). We can override this capability and fall back on AUDIO_LAYOUT based selection by setting AUDIO_LAYOUT_SELECT to _SW_BASED.

If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in the project specification, this field has no meaning, and only AUDIO_LAYOUT field will take effect.

SAMPLE_FLAT: SAMPLE_FLAT controls the values of (HB2[3:0]) of the Audio Sample Packet Header and should be _CLR. See Section 5.3.4 of the HDMI specification.

MAX_AC_PACKET: Set MAX_AC_PACKET to the maximum number of 32-pixel packets that will fit in the horizontal blanking interval. This controls the maximum number of audio packets, ACR packets, GCP packets, InfoFrames, etc. that will be sent during the horizontal blanking period.

MAX_AC_PACKET <= Floor[((HBLANK-REKEY-18)/32)]

CT_SELECT: Diagnostic workaround bit to decide whether the value of "coding type" will be hardware based or software based. The CT field is required in the construction of Audio InfoFrame packet (HDMI Specification, section 8.2.2). By default, the CT field is constructed by software. However, by setting this bit, the CT field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

CC_SELECT: Diagnostic workaround bit to decide whether the value of "channel count" will be hardware based or software based. The CC field is required in the construction of Audio InfoFrame packet (HDMI Specification, section 8.2.2). By default, the CC field is constructed by software. However,by setting this bit, the CC field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

SF_SELECT: Diagnostic workaround bit to decide whether the value of "sampling frequency" will be hardware based or software based. The SF field is required in the construction of Audio InfoFrame packet (HDMI Specification, section 8.2.2). By default, the SF field is constructed by software. However, by setting this bit, the SF field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

SS_SELECT: Diagnostic workaround bit to decide whether the value of "sample size" will be hardware based or software based. The SS field is required in the construction of Audio InfoFrame packet (HDMI Specification, section 8.2.2). By default, the SS field is constructed by software. However, by setting this bit, the SS field in the construction of the Audio InfoFrame

packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

CA_SELECT: Diagnostic workaround bit to decide whether the value of "channel allocation" will be hardware based or software based. The CA field is required in the construction of Audio InfoFrame packet (HDMI Specification, section 8.2.2). By default, the CA field is constructed by software. However, by setting this bit, the CA field in the construction of the Audio InfoFrame packet comes as a sideband signal from the HDA codec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field is reserved.

ENABLE: Set to _YES to enable HDMI for this head. Set to _NO to disable HDMI for this head.

Offset: 0x44 │ Byte Offset: 0x110 │ Read/Write: R/W │ Reset: 0x00020038 (0bx0x00000xxx00010xxx0x0x0x0111000)

Bit	Reset	Description
30	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
28	SW	CA_SELECT: 0 = SW 1 = HW
27	SW	SS_SELECT: 0 = SW 1 = HW
26	SW	SF_SELECT: 0 = SW 1 = HW
25	SW	CC_SELECT: 0 = SW 1 = HW
24	SW	CT_SELECT: 0 = SW 1 = HW
20:16	0x2	MAX_AC_PACKET
12	CLR	SAMPLE_FLAT: 0 = CLR 1 = SET
10	HW_BASED	AUDIO_LAYOUT_SELECT: 0 = HW_BASED 1 = SW_BASED
8	LAYOUT_2CH	AUDIO_LAYOUT: 0 = LAYOUT_2CH 1 = LAYOUT_8CH
6:0	0x38	REKEY

28.7.70 HDMI_NV_PDISP_HDMI_VSYNC_KEEPOUT_0

HDMI_VSYNC_KEEPOUT

Defines the start and end of the VSYNC keepout period where HDMI packets should not be sent. This is defined in chapter 2.7 the HDCP 1.1 specification.

END: Defines the end of the keepout period. This is measured in pixels after the active edge of VSYNC. This is defined in the specification and should not need to be written beyond its initialized value.

START: Defines the start of the keepout period. This is measured in pixels after the active edge of VSYNC. This is defined in the specification and should not need to be written beyond its initialized value.

ENABLE: When set to _YES, the keepout window is respected and no HDMI packets are sent in the period of time between START and END. This bit should be set to _YES. When set to _NO, the keepout window is ignored and HDMI packets may be sent regardless of the keepout window.

Offset: 0x45 │ Byte Offset: 0x114 │ Read/Write: R/W │ Reset: 0x819a028a (0b1xxxxx0110011010xxxxxx1010001010)

Bit	Reset	Description
31	YES	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
25:16	0x19a	START
9:0	0x28a	END

28.7.71 HDMI_NV_PDISP_HDMI_VSYNC_WINDOW_0

HDMI_VSYNC_WINDOW

Defines the start and end of the window of opportunity where the HDCP EESS signaling occurs. This is defined in chapter 2.7 of the HDCP 1.1 specification.

END: Defines the end of the window of opportunity. This is measured in pixels after the active edge of VSYNC. This is defined in the specification and should not need to be written beyond its initialized value.

START: Defines the start of the window of opportunity. This is measured in pixels after the active edge of VSYNC. This is defined in the specification and should not need to be written beyond its initialized value.

ENABLE: _YES will allow EESS signaling during the window of opportunity. _NO will prevent EESS signaling.

Offset: 0x46 │ Byte Offset: 0x118 │ Read/Write: R/W │ Reset: 0x82000210 (0b1xxxxx1000000000xxxxxx1000010000)

Bit	Reset	Description
31	YES	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
25:16	0x200	START
9:0	0x210	END

28.7.72 HDMI_NV_PDISP_HDMI_GCP_CTRL_0

The following registers are used to send a single-byte packet intended for sending the general control packet as described in section 5.3.6 of the HDMI specification. This control packet is used to control the AVMUTE flag.

HDMI_GCP_CTRL

This register controls the frequency and generation of GCP packets. The fields of this register are identical to HDMI_AUDIO_INFOFRAME_CTRL.

All four subpackets contain the same GCP packet.

Offset: 0x47 │ Byte Offset: 0x11c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxx0xxx0)

Bit	Reset	Description
8	DIS	SINGLE: 0 = DIS 1 = EN
4	DIS	OTHER: 0 = DIS 1 = EN
0	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

28.7.73 HDMI_NV_PDISP_HDMI_GCP_STATUS_0

HDMI_GCP_STATUS

The SENT bit will be set to _DONE, after the first packet is sent. After the ENABLE bit in INFOFRAME_CTRL is set to _NO, the SENT bit will be set to _WAITING after the start of the next frame to indicate that it is safe to change the contents of the packet header and subpacket registers.

Offset: 0x48 │ Byte Offset: 0x120 │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

28.7.74 HDMI_NV_PDISP_HDMI_GCP_SUBPACK_0

HDMI_GCP_SUBPACK

This register should be written with the contents of the general control packet.

See chapter 5.3.6 of the HDMI specification for more information.

Offset: 0x49 │ Byte Offset: 0x124 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxx00000000000000000000001)

Bit	Reset	Description
23:16	0x0	SB2
15:8	0x0	SB1
7:0	0x1	SB0: 1 = SET_AVMUTE 16 = CLR_AVMUTE

28.7.75 HDMI_NV_PDISP_HDMI_CHANNEL_STATUS1_0

HDMI_CHANNEL_STATUS1

This register is used for debug purposes only. Normally, the channel status bits encoded in the SPDIF sample stream are passed directly into the HDMI audio packet. HDMI_CHANNEL_STATUS1/2 can be used to override these bits with user defined values.

If STATUS2_ENABLE is set to _YES, then the 40-bit contents of CHANNEL_STATUS2/1 is inserted into the Cl and Cr bits of the audio sample packets for audio frames 0-39. Cl and Cr contain the channel status bits. See chapter 5.3.4 of the HDMI specification for information on Cr and Cl.

The fields in SPDIF_CHN_STATUS1/2 correspond to the fields in CHANNEL_STATUS1/2 for all values except for the ABCDM field. The values read from SPDIF_CHN_STATUS1/2 can be passed directly into the fields for CHANNEL_STATUS1/2 without any manipulation. This could be used if the user only needs to force one of the fields to a specific value and leave the rest untouched. Information on the channel status bits can be found in Chapter 5 of the IEC60958-3 specification.

ABCDM: This is the first byte of the channel status information. These bits have several meanings. See chapter 5.2.1 of IEC60958-3 for more information.

CODE: This byte defines the Category Code of the input device. This code indicates what type of equipment is generating the input. See 5.3.1 of the IEC60958-3 specification for the values of codes.

SOURCE: Source number of the audio.

CHANNEL: Channel number of the audio. The channel number inserted into channel 2 will be STATUS1_CHANNEL + 1

SFREQ: The reported sampling frequency of the audio stream.

ACCURACY: Transmitter Clock accuracy.

- LVL1 High accuracy mode: Transmitted sampling frequency is within a tolerance of +/-50*10^-6
- LVL2 Normal accuracy mode: Transmitted sampling frequency is within a tolerance of +/-1000*10^-6
- LVL3 Variable pitch shifted clock mode: Only specially designed receivers can receive the signal in this mode.
- OTHER: Interface frame rate not matched to sampling frequency: This is for all other modes that do not have an embedded sampling frequency clock.

Offset: 0x4a │ Byte Offset: 0x128 │ Read/Write: R/W │ Reset: 0xf2ffffff (0b11110010111111111111111111111111)

Bit	Reset	Description
31:28	0xf	ACCURACY: 1 = LVL1 0 = LVL2 2 = LVL3 3 = OTHER
27:24	0x2	SFREQ: 1 = UNDEFINED
23:20	0xf	CHANNEL
19:16	0xf	SOURCE
15:8	0xff	CODE
7:0	0xff	ABCDM

28.7.76 HDMI_NV_PDISP_HDMI_CHANNEL_STATUS2_0

HDMI_CHANNEL_STATUS2

MAX_LENGTH: Reports if the maximum audio sample word length is 20 bits or 24 bits.

LENGTH: Reports the audio sample word length of this block. The value of these bits is dependent on the value of MAX_LENGTH

If MAX_LENGTH = _20 then refer to the MAX20_* defines.

If MAX_LENGTH = _24 then refer to the MAX24_* defines.

All other bit combinations are reserved.

ORIGINAL: Defines the original sampling frequency of the audio stream.

ENABLE: _YES: override the channel status data with the value of these registers. _NO: Send the original channel status data

Offset: 0x4b │ Byte Offset: 0x12c │ Read/Write: R/W │ Reset: 0x000000f1 (0b0xxxxxxxxxxxxxxxxxxxxxxxx11110001)

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
7:4	0xf	ORIGINAL: 0 = UNDEFINED
3:1	0x0	LENGTH: 0 = MAX20_UNDEF 1 = MAX20_16BITS 2 = MAX20_18BITS 4 = MAX20_19BITS 5 = MAX20_20BITS 6 = MAX20_17BITS 0 = MAX24_UNDEF 1 = MAX24_20BITS 2 = MAX24_22BITS 4 = MAX24_23BITS 5 = MAX24_24BITS 6 = MAX24_21BITS
0	0x1	MAX_LENGTH

28.7.77 HDMI_NV_PDISP_HDMI_EMU0_0

HDMI_EMU0 and 1 do indirect addressing. EMU0 has the address, and EMU1 has the data.

To write:

1. Write the data register with data.

2. Write the address register with bit 31=1, and the address in bits 15:0.

3. Write the address register with bit 31=0.

To read:

1. Write the address register with bit 31=0 and the address in bits 15:0.

2. Read back the address register to introduce delay allowing the read data to propagate.

3. Read the data register.

Offset: 0x4c │ Byte Offset: 0x130 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	regx

28.7.78 HDMI_NV_PDISP_HDMI_EMU1_0

Offset: 0x4d │ Byte Offset: 0x134 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	regx

28.7.79 HDMI_NV_PDISP_HDMI_EMU1_RDATA_0

Offset: 0x4e │ Byte Offset: 0x138 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	0x0	rdata

28.7.80 HDMI_NV_PDISP_HDMI_SPARE_0

HDMI_SPARE

HW_CTS - If this is set to _ENABLE, Hardware measured CTS will be enabled. This method is more accurate than using software controlled CTS. The HDMI block will count the number of TMDS clocks that occur during the interval 1/(128 * audio sample rate) and use the value as CTS and issue the ACR Packet at each interval. When hardware measured CTS is enabled, all ENABLE fields of ACR_CTRL must be set to _NO. Also, ACR_0441_SUBPACK_HIGH_ENABLE must be set to _YES.

ACR_0441_SUBPACK_HIGH_N should be written with the value of N for the current sampling frequency.

The N value must also be written into AUDIO_N_VALUE (See the AUDIO register section for more details).

The value of N can be determined from the current audio sampling frequency, the current pixel clock rate, and tables 7-1. 7-2, and 7-3 in the HDMI specification. In most cases the value in the "Other" row can be used.

FORCE_SW_CTS: When HW_CTS = ENABLE, it uses measured CTS. However, when the audio clock / video clock ratio is known, this is undesirable. Currently, HW_CTS cannot be disabled.

When FORCE_SW_CTS=ENABLE along with HW_CTS=ENABLE, the HW_CTS is used to control transmission of ACR packets, but the CTS is programmed by software in the ACR_0441_SUBPACK_LOW registers.

SUPRESS_SP_B: Default SUPRESS_SP_B=0 is to disable B bits in audio sample packets, for non-present samples. Setting SUPRESS_SP_B=1 restores the old behavior.

CTS_RESET_VAL: When an ACR packet is sent, the CTS counter is reset to the value in this field. This should be set to the INIT value of 1

ACR_PRIORITY: This controls the priority of the ACR packet with respect to Audio Sample packets. This should be set to _HIGH.

LOW: ACR packets will have lower priority than Audio Sample packets

HIGH: ACR packets will have higher priority than Audio Sample packets

Software needs to make sure these registers are set when audio is active:

- HDMI_SPARE_HW_CTS = _HW_CTS_ENABLE
- HDMI_SPARE_CTS_RESET_VAL = 1
- HDMI_SPARE_ACR_PRIORITY = _HIGH
- ACR_CTRL_*_ENABLE = _NO
- ACR_0441_SUBPACK_HIGH_ENABLE = _YES

- ACR_0441_SUBPACK_HIGH_N = N
- AUDIO_N_VALUE = N
- ACR_0441_SUBPACK_LOW_SB1 = _USE_HW_CTS_VAL

HDMI_SPARE: 0:0 rw HW_CTS init=DISABLE

enum (DISABLE,ENABLE)

1:1 rw FORCE_SW_CTS init=DISABLE

enum (DISABLE,ENABLE)

2:2 rw SUPRESS_SP_B init=SUPRESS

enum (SUPRESS, KEEP)

18:16 rw CTS_RESET_VAL init=1

31:31 rw ACR_PRIORITY init=HIGH

enum (HIGH, LOW)

Offset: 0x4f │ Byte Offset: 0x13c │ Read/Write: R/W │ Reset: 0x00010000 (0b00000000000000010000000000000000)

Bit	Reset	Description
31:0	0x10000	Reserved.

28.7.81 HDMI_NV_PDISP_HDMI_SPDIF_CHN_STATUS1_0

HDMI_SPDIF_CHN_STATUS1

HDMI_SPDIF_CHN_STATUS1/2 contains the value of the channel status bits extracted from the incoming S/PDIF audio data stream. See IEC60958-3 chapter 5 for more information on these fields.

USE Specifies consumer use (CONSUMER) or professional (PRO) use of the channel status block. SPDIF_CHN_STATUS1/2 assumes consumer use.

TYPE: Specifies the type of data the audio word represents.

- PCM: Audio sample word represents linear PCM samples
- OTHER: Audio sample word is something other than liner PCM (i.e., compressed audio)

COPYRIGHT: Copyright status of the audio.

- YES: Copyright is asserted
- NO: Copyright is not asserted

D: The values of these bits have different meanings based on the value of _TYPE.

If _TYPE==_PCM, then:

- NO_PREEMPHASIS: 2 audio channels without pre-emphasis
- PREEMPHASIS: 2 audio channels with 50 microseconds/15 microseconds pre-emphasis

All other states are reserved for future use.

MODE Defines one of four possible channel status formats for bytes 1-23 of channel status. Currently only the value "0" is defined. This format is assumed for the SPDIF_CHN_STATUS1/2.

CODE: This byte defines the Category Code of the input device. This code indicates what type of equipment is generating the input. See 5.3.1 of the IEC60958-3 specification for the values of codes.

SOURCE: Source number of the audio. If this value is zero then no source number was reported.

CHANNEL: Channel number of the audio. This reports the channel number reported for the first subframe of audio. If this value is zero then the channel number was not reported.

SFREQ: This is the reported sampling frequency of the input audio stream. The value UNDEFINED means that the sampling frequency was not indicated by the audio stream.

ACCURACY: Transmitter Clock accuracy.

- LVL1 High accuracy mode: Transmitted sampling frequency is within a tolerance of +/-50*10^-6
- LVL2 Normal accuracy mode: Transmitted sampling frequency is within a tolerance of +/-1000*10^-6
- LVL3 Variable pitch shifted clock mode: Only specially designed receivers can receive the signal in this mode
- OTHER: Interface frame rate not matched to sampling frequency: This is for all other modes that do not have an embedded sampling frequency clock.

Offset: 0x50 │ Byte Offset: 0x140 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:28	X	ACCURACY: 1 = LVL1 0 = LVL2 2 = LVL3 3 = OTHER
27:24	X	SFREQ: 1 = UNDEFINED
23:20	X	CHANNEL: 0 = UNDEFINED
19:16	X	SOURCE: 0 = UNDEFINED
15:8	X	CODE
7:6	X	MODE
5:3	X	D: 0 = NO_PREEMPHASIS 1 = PREEMPHASIS
2	X	COPYRIGHT: 0 = YES 1 = NO
1	X	TYPE: 0 = PCM 1 = OTHER
0	X	USE: 0 = CONSUMER 1 = PRO

28.7.82 HDMI_NV_PDISP_HDMI_SPDIF_CHN_STATUS2_0

HDMI_SPDIF_CHN_STATUS2

MAX_LENGTH: Reports if the maximum audio sample word length is 20 bits or 24 bits.

LENGTH: Reports the audio sample word length of this block. The value of these bits is dependent on the value of MAX_LENGTH

- If MAX_LENGTH=_20 then refer to the MAX20_* defines.

- If MAX_LENGTH=_24 then refer to the MAX24_* defines.

All other bit combinations are reserved.

ORIGINAL: The original sampling frequency of the audio data. UNDEFINED indicates that the original sampling frequency was not defined

Offset: 0x51 │ Byte Offset: 0x144 │ Read/Write: RO │ Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:4	X	ORIGINAL: 0 = UNDEFINED
3:1	X	LENGTH: 0 = MAX20_UNDEF 1 = MAX20_16BITS 2 = MAX20_18BITS 4 = MAX20_19BITS 5 = MAX20_20BITS 6 = MAX20_17BITS 0 = MAX24_UNDEF 1 = MAX24_20BITS 2 = MAX24_22BITS 4 = MAX24_23BITS 5 = MAX24_24BITS 6 = MAX24_21BITS
0	X	MAX_LENGTH

28.7.83 HDMI_NV_PDISP_CRC_CONTROL_0

CRC_CONTROL

The main CRC enable bit flows along the pipeline. This register controls the injection of this bit at the top of the pipeline. This register is double-buffered. The active value is updated at start of each frame from this 'ARM' register.

Other registers such as SOR_STATE2.ASY_CRCMODE and SOR_*CRC* control the actual CRC logic, once enabled.

ARM_CRC_CONTROL enables or disables computation of CRC, effective at the start of next frame.

Offset: 0x96 │ Byte Offset: 0x258 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	NO	ARM_CRC_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

28.7.84 HDMI_NV_PDISP_INPUT_CONTROL_0

INPUT_CONTROL

HDMI_SRC_SELECT selects from which of the two display units to take input. This register should be changed only when HDMI is idle.

ARM_RGB_RANGE controls whether R/G/B values of 0 and 255 are permitted (FULL), or removed by clamping to [1,254] (LIMITED).

According to the EIA/CEA-861-B and HDMI specifications, the 640x480 VGA mode uses FULL, and all others use LIMITED.

Note that this does not scale the video or change the black and white points (VGA is [0,255], others are [16,235]). That must be done, if necessary, in display or at the source.

See the HDMI 1.2a specification, section 6.6.

This register is double-buffered and will take effect on next frame boundary.

Offset: 0x97 │ Byte Offset: 0x25c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00) | Default: 0x00000001

Bit	Reset	Default	Description
1	FULL	NONE	ARM_VIDEO_RANGE: 0 = FULL 1 = LIMITED
0	DISPLAY	DISPLAYB	HDMI_SRC_SELECT: 0 = DISPLAY 1 = DISPLAYB

28.7.85 HDMI_NV_PDISP_SCRATCH_0

SCRATCH

This register is not used by hardware. It is available for software to store the state or for testing purposes.

Offset: 0x98 │ Byte Offset: 0x260 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DATA

28.7.86 HDMI_NV_PDISP_PE_CURRENT_0

NV_PDISP_PE_CURRENT

- PE_CURRENT3_3.0
- PE_CURRENT2_3.0
- PE_CURRENT1_3.0
- PE_CURRENT0_3.0

Individual lane Pre-emphasis Current Control (4 bits per lane). The mapping of register value to current is as follows:

- 0000: 0.0mA
- 0001: 1.0mA
- 0010: 2.0mA
- 0011: 3.0mA
- 0100: 4.0mA
- 0101: 5.0mA
- 0110: 6.0mA
- 0111: 7.0mA
- 1000: 8.0mA
- 1001: 9.0mA
- 1010: 10mA
- 1011: 11mA

- 1100: 12mA
- 1101: 13mA
- 1110: 14mA
- 1111: 15mA

See related pre-emphasis control registers above in the NV_PDISP_SOR_PLL0 register.

Offset: 0x99 │ Byte Offset: 0x264 │ Read/Write: R/W │ Reset: 0x08080808 (0bxxxx1000xxxx1000xxxx1000xxxx1000) | Default: 0x00000000

Bit	Reset	SW Default	Description
27:24	0x8	0x0	PE_CURRENT3
19:16	0x8	0x0	PE_CURRENT2
11:8	0x8	0x0	PE_CURRENT1
3:0	0x8	0x0	PE_CURRENT0

28.7.87 HDMI_NV_PDISP_KEY_CTRL_0

HDCP KEY SRAM Register Control

This is the register space for the HDCP ROM interface control register. This register controls writing the contents of the hdcp_key_data bus (56 bits) into the local key store. The keys are decoded in the Crypto block, and then the key data is presented on the bus.

LOCAL: If this bit is ENABLED the on-chip HDCP key store will be used by the HDCP encryption block. If DISABLED, external Crypto-ROM will be used.

AUTOINC: If this bit is ENABLED, the address written to by the WRITE16 function will auto-increment after the operation is complete. This is the normal operating mode.

WRITE16: If this bit is set to TRIGGER, the HDCP keys module will write all 16 bytes of data from the hdcp_key_data bus (sourced by the CD module) into the local key store. The bytes are written into the store at contiguous bytes pointed to by the ADDRESS field in auto-increment mode, or contiguous bytes pointed to by the LOAD_ADDRESS field otherwise. Poll this bit until it reports DONE to ensure the write is complete.

PKEY_REQUEST_RELOAD: Requests that the private key be requested again from KFUSE. Will autoclear to zero as soon as the requested transfer of the key begins. ,Only PKEY_LOADED will indicate when the key is ready for use.

PKEY_LOADED: Indicates that the private key value has been received from KFUSE and is ready for use.

LOAD_ADDRESS: For the WRITE16 function, this field selects the start byte address of the contiguous locations in the local key store to be written with the hdcp_key_data. This field is ignored if the AUTOINC bit is ENABLED. Addresses start at 0.

ADDRESS: This read-only field reports the next byte address in the local key store to be written to once the TRIGGER bits are DONE. For the WRITE16 operation, the address will increment by 16. Addresses start at 0.

Typical operation is as follows:

1. Write to the register with LOCAL_ENABLED, AUTOINC_DISABLED, WRITE5_INIT, WRITE7_INIT, LOAD_ADDRESS_ZERO.
2. Write the downstream KSV key data into the CD module and decrypt the key (see dev_cd.ref - HDCP_KEY_REG(i) and NV_PCIPHER_CTL1).
3. Write to the register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE5_TRIGGER, WRITE7_INIT, LOAD_ADDRESS_ZERO. Poll for WRITE5_DONE.
4. Write the first downstream key data into the CD module and decrypt the key (see dev_cd.ref - HDCP_KEY_REG(i) and NV_PCIPHER_CTL1).

5. Write to the register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE7_TRIGGER, WRITE7_INIT, LOAD_ADDRESS_ZERO. Poll for WRITE7_DONE.
6. Repeat steps 4 and 5 39 more times. There are always 40 keys and one KSV value.

If upstream key authentication is required do the following:

7. Write the upstream KSV key data into the CD module and decrypt the key.
8. Write to the register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE5_TRIGGER, WRITE7_INIT, LOAD_ADDRESS_ZERO. Poll for WRITE5_DONE.
9. Write the first upstream key data into the CD module and decrypt the key (see dev_cd.ref - HDCP_KEY_REG(i) and NV_PCIPHER_CTL1).
10. Write to the register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE7_TRIGGER, WRITE7_INIT, LOAD_ADDRESS_ZERO. Poll for WRITE7_DONE.
11. Repeat steps 9 and 10 39 more times. There are always 40 keys and one KSV value.

This will leave the store as follows (the required format):

- Byte 0 to Byte 4: downstream KSV
- Byte 5 to Byte 11: 1st downstream key
- Byte 12 to Byte 18: 2nd downstream key

  ...
- Byte 278 to Byte 284: 40th downstream key
- Byte 285 to Byte 289: upstream KSV
- Byte 290 to Byte 296: 1st upstream key

  ...
- Byte 563 to Byte 569: 40'th upstream key

Offset: 0x9a │ Byte Offset: 0x268 │ Read/Write: R/W │ Reset: 0xXXX000X0 (0bxxxxxxxxxx0000000000xxxxxx00xx00)

Bit	R/W	Reset	Description
31:22	RO	X	ADDRESS
21:12	RW	0x0	LOAD_ADDRESS
6	RO	X	PKEY_LOADED: 0 = FALSE 1 = TRUE
5	RW	IDLE	PKEY_REQUEST_RELOAD: 0 = IDLE 1 = TRIGGER
4	RW	DONE	WRITE16: 0 = DONE 1 = TRIGGER 1 = PENDING
1	RW	DISABLED	AUTOINC: 0 = DISABLED 1 = ENABLED
0	RW	DISABLED	LOCAL_KEYS: 0 = DISABLED 1 = ENABLED

28.7.88 HDMI_NV_PDISP_KEY_DEBUG0_0

Offset: 0x9b | Byte Offset: 0x26c | Read/Write: R/W | Reset: 0x000000X0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xxx0)

Bit	R/W	Reset	Description
6	RO	X	CHECKSUMCMP_HIGH: 0 = MISMATCH 1 = MATCH
5	RO	X	CHECKSUMCMP_LOW: 0 = MISMATCH 1 = MATCH
4	RW	DONE	CHECKSUM: 0 = DONE 1 = TRIGGER 1 = PENDING
0	RW	DONE	SRAMCLEAR: 0 = DONE 1 = TRIGGER 1 = PENDING

28.7.89 HDMI_NV_PDISP_KEY_DEBUG1_0

Offset: 0x9c | Byte Offset: 0x270 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	CHECKSUMVAL_HIGH
15:0	0x0	CHECKSUMVAL_LOW

28.7.90 HDMI_NV_PDISP_KEY_DEBUG2_0

Offset: 0x9d | Byte Offset: 0x274 | Read/Write: R/W | Reset: 0xXXXXX000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xx0x)

Bit	Reset	Description
31:24	X	SRAMDATA
21:12	X	SRAMADDR
4	DONE	SRAMWRITE1: 0 = DONE 1 = TRIGGER 1 = PENDING
1	DISABLED	SRAMAUTOINC: 0 = DISABLED 1 = ENABLED

28.7.91 HDMI_NV_PDISP_KEY_HDCP_KEY_0_0

Offset: 0x9e | Byte Offset: 0x278 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	KEY_REG

28.7.92 HDMI_NV_PDISP_KEY_HDCP_KEY_1_0

Offset: 0x9f | Byte Offset: 0x27c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	KEY_REG

28.7.93 HDMI_NV_PDISP_KEY_HDCP_KEY_2_0

Offset: 0xa0 | Byte Offset: 0x280 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	KEY_REG

28.7.94 HDMI_NV_PDISP_KEY_HDCP_KEY_3_0

Offset: 0xa1 | Byte Offset: 0x284 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	KEY_REG

28.7.95 HDMI_NV_PDISP_KEY_HDCP_KEY_TRIG_0

Offset: 0xa2 | Byte Offset: 0x288 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxxx)

Bit	Reset	Description
8	IDLE	LOAD_HDCP_KEY: 0 = IDLE 1 = TRIGGER

28.7.96 HDMI_NV_PDISP_KEY_SKEY_INDEX_0

Sixteen (16) sets of AES keys are available. The convention is:

- 15: test/debug
- 0-14: production

Offset: 0xa3 | Byte Offset: 0x28c | Read/Write: WO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3:0	0x0	IDX_VALUE: 15 = TEST

28.7.97 HDMI_NV_PDISP_INT_STATUS_0

Sticky interrupt status, write 1 to clear

Interrupt support: Each interrupt event has 3 configuration states:

MASK	ENABLE	Result
x	0	nothing
0	1	INT_STATUS asserted
1	1	INT_STATUS asserted, and interrupt pin asserted

Offset: 0xcc │ Byte Offset: 0x330 │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3	X	SCRATCH: Software has written NV_PDISP_SCRATCH register (bit 31 changed)
2	X	CP_REQUEST: HDA Codec has written CP_REQUEST register
1	X	CODEC_SCRATCH1: HDA Codec has written CODEC_SCRATCH1 register (bit 31 changed)
0	X	CODEC_SCRATCH0: HDA Codec has written CODEC_SCRATCH0 register (bit 31 changed)

28.7.98 HDMI_NV_PDISP_INT_MASK_0

MASKED prevents the interrupt from asserting the HDMI interrupt pin to the CPU, but still allows the interrupt to appear in INT_STATUS. NOTMASKED allows the interrupt to assert, assuming INT_ENABLE is true also.

Offset: 0xcd │ Byte Offset: 0x334 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	MASKED	SCRATCH_MASK: 0 = MASKED 1 = NOTMASKED
2	MASKED	CP_REQUEST_MASK: 0 = MASKED 1 = NOTMASKED
1	MASKED	CODEC_SCRATCH1_MASK: 0 = MASKED 1 = NOTMASKED
0	MASKED	CODEC_SCRATCH0_MASK: 0 = MASKED 1 = NOTMASKED

28.7.99 HDMI_NV_PDISP_INT_ENABLE_0

ENABLE allows the event to appear in INT_STATUS, and allows the interrupt to signal the CPU, assuming INT_MASK=NOTMASKED.

Offset: 0xce │ Byte Offset: 0x338 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	DISABLE	SCRATCH_ENABLE: 0 = DISABLE 1 = ENABLE
2	DISABLE	CP_REQUEST_ENABLE: 0 = DISABLE 1 = ENABLE
1	DISABLE	CODEC_SCRATCH1_ENABLE: 0 = DISABLE 1 = ENABLE
0	DISABLE	CODEC_SCRATCH0_ENABLE: 0 = DISABLE 1 = ENABLE

28.8 Serial Output Resource Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

28.8.1 HDMI_NV_PDISP_SOR_PWR_0

This register contains bits that control the power state of the SOR. For simplicity, the sequencer will be used to perform all power control operations in the SOR (even in TMDS where the sequencing is relatively simple). This unifies the approach and makes it easier for the hardware and software to deal with the SOR's. At boot, the software must load and configure the SOR sequencer via the SOR_SEQ_CTL and SOR_SEQ_INST registers below. The sequencer must be properly programmed for power control to work.

NORMAL_STATE: Sets the normal operating state. There are two choices: powered up and powered down. To change this register, software should load in the new value and write SETTING_NEW=TRIGGER. Once the change has been successfully completed, SETTING_NEW will indicate DONE. This is the state that software will want to control in order to power up and down the interface.

NORMAL_START: The execution of the power up and power down sequence can be modified somewhat by choosing to use either the NORM or ALT program entry point.

SAFE_STATE: Sets the safe operating state. There are only two choices: powered up and powered down. To change this register, software should load in the new value and write SETTING_NEW=TRIGGER. Once the change has been successfully completed, the SETTING_NEW will indicate DONE. The execution of the power up and power down sequence can be modified somewhat by choosing to use either the standard or the alternate program entry point. This is the state that the hardware will use whenever it initiates the mode switch/shutdown procedure for HDMI. In general, this should be set to STATE_PD and left there always.

SAFE_START: The execution of the power up and power down sequence can be modified somewhat by choosing to use either the NORM or ALT program entry point.

HALT_DELAY: Once the sequencer gets to the instruction with the HALT=1, the program is essentially complete; that is, the attached unit is powered up or down as was requested. Some panels, however, have a minimum time before their state can be changed again. If the halt instruction has a non-zero delay, this bit will be set while that time expires.

MODE: The currently active state, normal or safe. After reset, the MODE is always SAFE.

SETTING_NEW: This bit is used to trigger a new setting of power mode to take effect. The typical procedure might be something like:

1. Make sure SETTING_NEW==DONE, i.e., not already an outstanding change request. If there is an outstanding change request, software must wait for it to complete.

2. Update the NORMAL and SAFE power mode fields.

3. Write SETTING_NEW=TRIGGER.

4. Poll for SETTING_NEW=DONE. If it does not happen within one frame time, then software may opt to accelerate the change by writing SOR_SEQ_CTL SWITCH=FORCE (be careful not to disturb other settings in that register!).

Usage: boot / initialization / mode switch / normal operation / shutdown

Offset: 0x55 │ Byte Offset: 0x154 │ Read/Write: R/W │ Reset: 0xXX000000 (0b0xxxxxxxxxxxxx00xxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
31	RW	0x0	SETTING_NEW: 0 = DONE 1 = PENDING 1 = TRIGGER

Bit	R/W	Reset	Description
28	RO	X	MODE: 0 = NORMAL 1 = SAFE
24	RO	X	HALT_DELAY: 0 = DONE 1 = ACTIVE
17	RW	0x0	SAFE_START: 0 = NORMAL 1 = ALT
16	RW	0x0	SAFE_STATE: 0 = PD 1 = PU
1	RW	0x0	NORMAL_START: 0 = NORMAL 1 = ALT
0	RW	0x0	NORMAL_STATE: The normal state depends on the power sequencing state. NORMAL_STATE=1 is correct for powered-up HDMI. 0 is for the powered-down / reset state. 0 = PD 1 = PU

28.8.2 HDMI_NV_PDISP_SOR_TEST_0

This register contains control bits that configure certain aspects of testing mode of the SOR.

TEST_ENABLE: To enable testing:

- 0 = normal operation
- 1 = test mode

In test mode test_fastclkint and test_loadpulse signals from the core are used in place of the internally generated signals.

INVD: Invert the data. Note that combining INVD with the choice of DSRC and TPAT below permits ramp up, ramp down, walking ones, walking zeros, all ones, all zeros.

ACT_HEAD_OPMODE: Report the current OR operating mode.

ATTACHED: Report whether the OR is currently attached to a head.

DSRC: When DSRC=NORMAL, the serializers behave normally and use the encoded RGB data. When DSRC=DEBUG, the serializers load from DEBUGA0, DEBUGA1, DEBUGB0, and DEBUGB1 bits instead of the normal data coming down the pipe. When DSRC=TGEN, the data is taken from the built in test generator.

TPAT: Selects the test generator pattern. When test generator is running, H will be high for 1 in 4 repetitions (of duration 1024 clocks), V will be high for 4 in 16 repetitions coincident with H. Only operates when test mode is enabled.

LO: force all 0s

TDAT: use hardware fixed value

RAMP: test generator output ramps

WALK: test generator output is a walking one

MAXSTEP: 0, 1023, ...

MINSTEP: 511, 512, ...

CRC: The source of the data for CRC computation (for verification) can be either before the high-speed serializer logic, or after the high-speed serializer/deserializer logic.

TESTMUX[7:0]: Test MUX select - output seen on PROBE

Usage: Debug

Offset: 0x56 │ Byte Offset: 0x158 │ Read/Write: R/W │ Reset: 0x00800X00 (0b000000001000xx00xxxxxxxxx0xxxx0x)

Bit	R/W	Reset	Description
31:24	RW	0x0	TESTMUX: 0 = AVSS 2 = CLOCKIN 4 = PLL_VOL 8 = SLOWCLKINT 16 = AVDD 32 = VDDREG 64 = REGREF_VDDREG 128 = REGREF_AVDD
23	RW	0x1	CRC: 0 = PRE_SERIALIZE 1 = POST_DESERIALIZE
22:20	RW	0x0	TPAT: 0 = LO 1 = TDAT 2 = RAMP 3 = WALK 4 = MAXSTEP 5 = MINSTEP
17:16	RW	0x0	DSRC: 0 = NORMAL 1 = DEBUG 2 = TGEN
10	RO	X	ATTACHED: 0 = FALSE 1 = TRUE
9:8	RO	X	ACT_HEAD_OPMODE: 0 = SLEEP 1 = SNOOZE 2 = AWAKE
6	RW	0x0	INVD: 0 = DISABLE 1 = ENABLE
1	RW	0x0	TEST_ENABLE: 0 = DISABLE 1 = ENABLE

28.8.3 HDMI_NV_PDISP_SOR_PLL0_0

These registers configure the main SOR PLL and other frequency-dependent controls. The value loaded is a function of the pixel clock frequency, and whenever the pixel clock changes, these registers must be updated. In general, these registers will be updated during the second interrupt of a mode switch.

PWR: 1 = power down the TMDS PLL

PDBG: 1 = power down the bandgap

VCOPD: 1 = power down the VCO

PDPORT: 1 = power down the output drivers

RESISTORSEL: Selects the internal resistor (0) or external resistor (1).

PULLDOWN: Weak pull-down enable

- 1 = 2Kohm pulldown on all outputs.
- 0 = Weak pulldown disabled.

Note: pulldown is also controlled by the sequencer. the pulldown is derived from an OR of:

NV_PDISP_SOR_PLL0_PULLDOWN and the sequencer control of pulldown. This priv register field does not read the final pull-down value. In other words if software writes NV_PDISP_SOR_PLL0_PULLDOWN to DISABLED and sequencer pulldown is enabled, then an NV_PDISP_SOR_PLL0_PULLDOWN read will return DISABLED (while pulldown might be ENABLED due to sequencer control).

VCOCAP[3:0]: Selects the VCO capacitor and adjusts ring oscillator inter-stage load.

FILTER[3:0]: Selects the loop filter and adjusts the filter resistor value.

ICHPMP[3:0]: Specifies additions to the charge pump current in steps of 0.375 µA.

TMDS_TERM: This bit is used to enable termination. Termination is only used in TMDS mode of operation, and may not be needed at lower operating frequencies (in which case, disabling it saves power).

TERMADJ[3:0]: Termination resistance control.

LOADADJ[3:0]: Load pulse position adjust.

AUX0-AUX7: Most of these bits currently have no assigned function, but are provided to permit control of possible future features of the macro.

- AUX0: No function
- AUX1: No function
- AUX2: No function
- AUX3: Rotate green channel by 1 bit to reduce TMDS EMI
- AUX4: No function
- AUX5: No function
- AUX6: No function
- AUX7: No function

TMDS_ANALOG_X4_HP_B revision:

BG_V17_S[3:0]: Bandgap 1.7V output voltage control

TX_REG_LOAD[1:0]: TX regulator default loading

- 00: 0.5mA (default)
- 01: 1.0mA
- 10: 1.5mA
- 11: 2.0mA

PE_EN: Pre-emphasis enable. PE_EN active only at 1080p, but turn off at 720p

- 0: Disable
- 1: Enable

HALF_FULL_PE: Pre-emphasis half bit time or full bit time (Note 2)

- 0: Half bit time (default)
- 1: Full bit time (for long trace)

S_D_PIN_PE: Pre-emphasis on one single pin or on differential pins (D+ and D-) (Note 2)

- 0: Single pin (default)
- 1: Differential pins

See also the NV_PDISP_PE_CURRENT register below.

Usage: boot / initialization / mode switch

Offset: 0x57 │ Byte Offset: 0x15c │ Read/Write: R/W │ Reset: 0x0200333f (0bxx000010xxxx000000110011xx111111)| Default: 0x01000000

Bit	Reset	Default	Description
29:28	0x0	NONE	TX_REG_LOAD: 0 = RESETV
27:24	0x2	0x1	ICHPMP: 2 = RESETV
19:16	0x0	NONE	FILTER: 0 = RESETV
15:12	0x3	NONE	BG_V17_S: 3 = RESETV
11:8	0x3	NONE	VCOCAP: 3 = RESETV
5	0x1	DISABLE	PULLDOWN: 0 = DISABLE 1 = ENABLE
4	0x1	NONE	RESISTORSEL: 0 = INT 1 = EXT 1 = RESETV
3	0x1	NONE	PDPORT: 0 = ON 1 = OFF
2	0x1	NONE	VCOPD: 0 = RESCIND 1 = ASSERT
1	0x1		PDBG: 0 = ON 1 = OFF
0	0x1		PWR: 0 = ON 1 = OFF

28.8.4 HDMI_NV_PDISP_SOR_PLL1_0

Usage: boot / initialization / mode switch

Offset: 0x58 │ Byte Offset: 0x160 │ Read/Write: R/W │ Reset: 0x00301200 (0bx000xxxx0011xxxxxxx10010xxxxxxxx)

Bit	Reset	Description
30	SINGLE	S_D_PIN_PE: 0 = SINGLE 1 = DIFFERENTIAL
29	HALF	HALF_FULL_PE: 0 = HALF 1 = FULL

Bit	Reset	Description
28	DISABLE	PE_EN: 0 = DISABLE 1 = ENABLE
23:20	0x3	LOADADJ: 0 = CENTER
12:9	0x9	TMDS_TERMADJ
8	0x0	TMDS_TERM: 0 = DISABLE 1 = ENABLE

28.8.5 HDMI_NV_PDISP_SOR_PLL2_0

Spare Registers for TMDS Control

AUX3: lane 1 rotation control

- 0: No rotation (Default)
- 1: Rotate right 1 bit

Usage: boot / initialization / mode switch

Offset: 0x59 │ Byte Offset: 0x164 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx00000000xxxxxxxxxxxxxxxx)

Bit	Reset	Description
23	0x0	AUX7
22	0x0	AUX6
21	0x0	AUX5
20	0x0	AUX4
19	0x0	AUX3
18	0x0	AUX2
17	0x0	AUX1
16	0x0	AUX0

28.8.6 HDMI_NV_PDISP_SOR_CSTM_0

The class permits the driver to select a number of operating modes for the SOR. Some of these modes (TMDS modes) are well defined and stable. Some modes may require customization by software before they will work. These registers are used for that customization. The fields available for customization are:

- PD_TXDA[3:0]: Bitwise control to power down the data pins of link A. Set to DISABLE to power down the pin.
- PD_TXDB[3:0]: Bitwise control to power down the data pins of link B. Set to DISABLE to power down the pin.
- PD_TXCA: Power down the clock pin of link A. Set to DISABLE to power down the pin.
- PD_TXCB: Power down the clock pin of link B. Set to DISABLE to power down the pin.

UPPER: Designates whether LVDS bank A is the upper, odd, or first pixel. Bank B is always set to !UPPER. The serial output from UPPER=1 will be Clock and A0-A3. The serial output from UPPER=0 (and DUALMODE) will be Clock and A4-A7. The default is bank A has UPPER=1, bank B has UPPER=0.

MODE[1:0]: Controls the digital output encoding applied to the data stream in custom mode and is only used for data muxing at the input of the SOR. This field does not control the TMDS macro.

- 0 = LVDS
- 1 = TMDS
- 2 = Reserved
- 3 = Reserved

LINKACTA, LINKACTB: Enables (1) or disables (0) the digital logic of links A and B

LVDS_EN: Output driver configuration for controlling the encoding of the data and the output common mode control. This does not control the internal clock dividers of the TMDS macro.

- 0 = TMDS
- 1 = LVDS

DUP_SYNC: This field only has an effect when in LVDS mode. When asserted in LVDS mode, it forces the link to use DE, HSYNC, and VSYNC for the encoding and to never use RES, CNTLE, and CNTLF. RES becomes DE. CNTLE becomes HSYNC. CNTLF becomes VSYNC.

NEW_MODE: For backwards compatibility, set register to 0 so the old mode is used. In old mode, for the second link in dual link mode, all the control bits are zeroed out except for VSYNC. When a new mode is used, none of the control bits of the second link for dual-link mode are zeroed out.

BALANCED: For MODE = LVDS, this enables balanced encoding. Balanced mode will selectively invert sets of bits in the serial stream in an attempt to keep the average DC value near zero. Default is unbalanced. Has no effect in other modes.

PLLDIV: Controls the internal clock dividers of the TMDS_MACRO by setting the feedback divider for the high-speed PLL.

- 0 = divide by 7 (LVDS)
- 1 = divide by 10 (TMDS)

ROTCLK[3:0]: Skews the TXC clock to come out earlier. By changing this register value, you can configure the skew between output data (TX data) and output clock (TXC). This field specifies the number of sclk cycles which the output clock should come out earlier than its normal phase. For TMDS, sclk is a 10x pixel clock, so ROTVAL should be between 0-9. For LVDS, sclk is a 7x pixel clock, so ROTVAL should be between 0-7.

ROTDAT[2:0]: Before encoding the 8 bits of each color channel, the 8 bits within each color channel can be right rotated. All color channels are rotated by the same amount. For example, if ROTDAT = 6, then input channel data (r7,r6,r5,r4,r3,r2,r1,r0} would become {r5,r4,r3,r2,r1,r1,r7,r6}.

ROTDAT should be between 0 and 7.

TMDS modes of operation are standard and stable. The table below summarizes the fixed values the hardware uses for the above fields for TMDS operating modes.

	SINGLE TMDS_A	SINGLE TMDS_B	SINGLE TMDS_AB	SINGLE TMDS	Dual TMDS
PD_TXDA[2:0]	0	7	0	0	0
PD_TXDA[3]	1	1	1	1	1
PD_TXDB[2:0]	7	0	0	0	0
PD_TXDB[3]	1	1	1	1	1
PD_TXCA	0	1	0	0	0
PD_TXCB	1	0	0	0	1
UPPER	1	1	1	1	1
MODE[1:0]	1	1	1	1	1
LINKACTA	1	0	1	1	1
LINKACTB	0	1	1	1	1
LVDS_EN	0	0	0	0	0

	SINGLE TMDS_A	SINGLE TMDS_B	SINGLE TMDS_AB	SINGLE TMDS	Dual TMDS
DUP_SYNC	0	0	0	0	0
NEW_MODE	0?	1	1	1	0?
BALANCED	0	0	0	0	0
PLLDIV	1	1	1	1	1
ROTCLK[3:0]	0	0	0	0	0
ROTDAT[2:0]	0	0	0	0	0

The first register defines the total custom mode. This is useful for defining possible new modes of operation as well as for test and characterization in the lab. Which is to say, this is a debug register set. Software should really never need to use it. Also, note that if LVDS_ONLY=TRUE, then this register cannot be used as the hardware will only permit Protocol=LVDS to be enabled. This register is intended to have the same format as the second register.

Usage: boot / initialization / mode switch

Offset: 0x5a │ Byte Offset: 0x168 │ Read/Write: R/W │ Reset: 0x0001c800 (0bx0000000xx0x000111001x0000000000) | Default: 0x02000000

Bit	Reset	Default	Description
30:28	0x0	NONE	ROTDAT: 0 = RESETV
27:24	0x0	0x2	ROTCLK: 0 = RESETV
21	0x0	NONE	PLLDIV: 0 = BY_7 1 = BY_10
19	0x0	NONE	BALANCED: 0 = DISABLE 1 = ENABLE
18	0x0	NONE	NEW_MODE: 0 = DISABLE 1 = ENABLE
17	0x0	NONE	DUP_SYNC: 0 = DISABLE 1 = ENABLE
16	0x1	NONE	LVDS_EN: 0 = DISABLE 1 = ENABLE
15	0x1	NONE	LINKACTB: 0 = DISABLE 1 = ENABLE
14	0x1	NONE	LINKACTA: 0 = DISABLE 1 = ENABLE
13:12	0x0	NONE	MODE: 0 = LVDS 1 = TMDS
11	0x1	NONE	UPPER: 1 = TRUE 0 = FALSE
9	0x0	NONE	PD_TXCB: 1 = DISABLE 0 = ENABLE
8	0x0	NONE	PD_TXCA:

Bit	Reset	Default	Description
			1 = DISABLE 0 = ENABLE
7	0x0	NONE	PD_TXDB_3: 1 = DISABLE 0 = ENABLE
6	0x0	NONE	PD_TXDB_2: 1 = DISABLE 0 = ENABLE
5	0x0	NONE	PD_TXDB_1: 1 = DISABLE 0 = ENABLE
4	0x0	NONE	PD_TXDB_0: 1 = DISABLE 0 = ENABLE
3	0x0	NONE	PD_TXDA_3: 1 = DISABLE 0 = ENABLE
2	0x0	NONE	PD_TXDA_2: 1 = DISABLE 0 = ENABLE
1	0x0	NONE	PD_TXDA_1: 1 = DISABLE 0 = ENABLE
0	0x0	NONE	PD_TXDA_0: 1 = DISABLE 0 = ENABLE

28.8.7 HDMI_NV_PDISP_SOR_LVDS_0

The second register defines the LVDS custom mode. This is the base from which all LVDS variants can be customized. This register is intended to have the same format as the first register.

Usage: boot / initialization / mode switch

Offset: 0x5b │ Byte Offset: 0x16c │ Read/Write: R/W │ Reset: 0x00XXXX0X (0bx0000000xxxx000x1xxxxx0x00000xxx)

Bit	R/W	Reset	Description
30:28	RW	0x0	ROTDAT: 0 = RESETV
27:24	RW	0x0	ROTCLK: 0 = RESETV
21	RO	X	PLLDIV: 0 = BY_7
19	RW	0x0	BALANCED: 0 = DISABLE 1 = ENABLE
18	RW	0x0	NEW_MODE: 0 = DISABLE 1 = ENABLE
17	RW	0x0	DUP_SYNC: 0 = DISABLE 1 = ENABLE
16	RO	X	LVDS_EN: 1 = ENABLE

Bit	R/W	Reset	Description
15	RW	0x1	LINKACTB: 0 = DISABLE 1 = ENABLE
14	RO	X	LINKACTA: 1 = ENABLE
13:12	RO	X	MODE: 0 = LVDS
11	RO	X	UPPER: 1 = TRUE 0 = FALSE
9	RW	0x0	PD_TXCB: 1 = DISABLE 0 = ENABLE
8	RO	X	PD_TXCA: 0 = ENABLE
7	RW	0x0	PD_TXDB_3: 1 = DISABLE 0 = ENABLE
6	RW	0x0	PD_TXDB_2: 1 = DISABLE 0 = ENABLE
5	RW	0x0	PD_TXDB_1: 1 = DISABLE 0 = ENABLE
4	RW	0x0	PD_TXDB_0: 1 = DISABLE 0 = ENABLE
3	RW	0x0	PD_TXDA_3: 1 = DISABLE 0 = ENABLE
2	RO	X	PD_TXDA_2: 0 = ENABLE
1	RO	X	PD_TXDA_1: 0 = ENABLE
0	RO	X	PD_TXDA_0: 0 = ENABLE

28.8.8 HDMI_NV_PDISP_SOR_CRCA_0

The following three registers are used to fetch CRC's when running VGA mode tests. One contains the valid bit, one contains the actual computed CRC, and the third contains the error (overrun bit). Proper use of these registers is as follows:

1) Poll SOR_CRCA for VALID == TRUE.

2) Reset SOR_CRCA by writing RESET to it.

3) Read SOR_CRCB to get the CRC

Usage: verification

Offset: 0x5c │ Byte Offset: 0x170 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	VALID: 0 = FALSE

Bit	Reset	Description
		1 = TRUE 1 = RESETV

28.8.9 HDMI_NV_PDISP_SOR_CRCB_0

Usage: verification

Offset: 0x5d │ Byte Offset: 0x174 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	CRC

28.8.10 HDMI_NV_PDISP_SOR_BLANK_0

This register can be used to override the SOR output resource pixels with blank data.

OVERRIDE: Setting this field to true will override the pixel bus from the RG and output black pixels instead.

TRANSITION: This field controls the timing of the output resource blank override. The choices are IMMEDIATE. The output resource will be blanked or restored to its previous output data immediately.

NEXT_VSYNC The output resource will be blanked or restored to its previous output data at the next vsync.

STATUS This read-only field returns BLANKED when the output resource is sending blank pixels forced by the OVERRIDE bit, otherwise it returns NOT_BLANKED.

Usage: boot / initialization / mode switch / normal operation

Offset: 0x5e │ Byte Offset: 0x178 │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	R/W	Reset	Description
2	RO	X	STATUS: 0 = NOT_BLANKED 1 = BLANKED
1	RW	0x0	TRANSITION: 0 = IMMEDIATE 1 = NEXT_VSYNC
0	RW	0x0	OVERRIDE: 0 = FALSE 1 = TRUE

28.8.11 HDMI_NV_PDISP_SOR_SEQ_CTL_0

Sequencer control registers for SOR. Up to three pins can be assigned to an SOR for use in controlling the power to an attached LVDS flat panel. The meaning of any particular pin and whether it is actually available to this SOR is controlled in either the host or PCB manager. In addition, the sequencer can override the individual link clock and data power control pins, thereby forcing them all into disabled or tristate mode and it can override the DE signal from the RG (for TMDS mode) to force the link to become inactive.

PU_PC: The program counter for the start of the power up program sequence. Always 0.

PU_PC_ALT: The alternate entry point into the power up program sequence. Defaults to the same value as PU_PC.

PD_PC: The program counter for the start of the power down program sequence. Defaults to 8, evenly dividing the available program space.

PD_PC_ALT: The alternate entry point into the power down program sequence. Defaults to the same default value as PD_PC.

PC: The current value of the program counter (useful for status and debug).

STATUS: Indicates if the sequencer is STOPPED or RUNNING.

SWITCH: If a particular sequencer instruction is waiting for vsync to arrive, writing this bit to FORCE will cause the wait condition to be satisfied immediately.

> **Note:** The sequencer can begin with arbitrary phase relative to the 1 µs timer that is used to advance the internal counters. Thus the actual time delay for the first event can be almost one microsecond less than what is requested. Subsequent instructions in a chain of events do transition on 1 µs boundaries, however. If a minimum specification of "x" µs is required, then it is best to program "x+1" µs of delay.

Usage: boot / initialization / mode switch

Offset: 0x5f │ Byte Offset: 0x17c │ Read/Write: R/W │ Reset: 0xX00X880X (0bx0xxxxxxxxxxxxxx100010000000xxxx)

Bit	R/W	Reset	Description
30	RW	WAIT	SWITCH: 0 = WAIT 1 = FORCE
28	RO	X	STATUS: 0 = STOPPED 1 = RUNNING
19:16	RO	X	PC
15:12	RW	0x8	PD_PC_ALT
11:8	RW	0x8	PD_PC
7:4	RW	0x0	PU_PC_ALT
3:0	RO	X	PU_PC

28.8.12 HDMI_NV_PDISP_SOR_SEQ_INST0_0

Usage: boot / initialization

Offset: 0x60 │ Byte Offset: 0x180 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000) | Default: 0x00802001

Bit	Reset	Default	Description
31	0x0	NONE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	NONE	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	NONE	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	NONE	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	NONE	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	NONE	BLANK_DE: 0 = NORMAL 1 = INACTIVE

Bit	Reset	Default	Description
25	0x0	NONE	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	NONE	TRISTATE_IOS: 0 is the correct value for power-up. 1 is the value when HDMI is off. 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	TRUE	DRIVE_PWM_OUT_LO: Power sequencer; verified with value 1 (TRUE). 0 = FALSE 1 = TRUE
22	0x0	NONE	PIN_B: 0 = LOW 1 = HIGH
21	0x0	NONE	PIN_A: 0 = LOW 1 = HIGH
15	0x1	NONE	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	VSYNC	WAIT_UNITS: Power sequencer; verified with WAIT_UNITS=2 (VSYNC). 0 = US 1 = MS 2 = VSYNC
9:0	0x0	0x1	WAIT_TIME: Power sequencer; verified with WAIT_TIME=1.

28.8.13 HDMI_NV_PDISP_SOR_SEQ_INST1_0

Usage: boot / initialization

Offset: 0x61 │ Byte Offset: 0x184 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS

Bit	Reset	Description
		1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.14 HDMI_NV_PDISP_SOR_SEQ_INST2_0

Usage: boot / initialization

Offset: 0x62 │ Byte Offset: 0x188 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B:

Bit	Reset	Description
		0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.15 HDMI_NV_PDISP_SOR_SEQ_INST3_0

Usage: boot / initialization

Offset: 0x63 │ Byte Offset: 0x18c │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH

Bit	Reset	Description
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.16 HDMI_NV_PDISP_SOR_SEQ_INST4_0

Usage: boot / initialization

Offset: 0x64 │ Byte Offset: 0x190 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS

Bit	Reset	Description
		2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.17 HDMI_NV_PDISP_SOR_SEQ_INST5_0

Usage: boot / initialization

Offset: 0x65 │ Byte Offset: 0x194 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.18 HDMI_NV_PDISP_SOR_SEQ_INST6_0

Usage: boot / initialization

Offset: 0x66 | Byte Offset: 0x198 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.19 HDMI_NV_PDISP_SOR_SEQ_INST7_0

Usage: boot / initialization

Offset: 0x67 | Byte Offset: 0x19c | Read/Write: R/W | Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.20 HDMI_NV_PDISP_SOR_SEQ_INST8_0

Usage: boot / initialization

Offset: 0x68 | Byte Offset: 0x1a0 | Read/Write: R/W | Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000) | Default: 0x00802001

Bit	Reset	Default	Description
31	0x0	NONE	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	NONE	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	NONE	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	NONE	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	NONE	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	NONE	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	NONE	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	NONE	TRISTATE_IOS: 0 is the correct value for power-up. 1 is the value when HDMI is off. 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	TRUE	DRIVE_PWM_OUT_LO: Power sequencer; verified with value 1 (TRUE). 0 = FALSE 1 = TRUE
22	0x0	NONE	PIN_B: 0 = LOW 1 = HIGH
21	0x0	NONE	PIN_A: 0 = LOW 1 = HIGH
15	0x1	NONE	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	VSYNC	WAIT_UNITS: Power sequencer; verified with WAIT_UNITS=2 (VSYNC) 0 = US 1 = MS 2 = VSYNC
9:0	0x0	0x1	WAIT_TIME: Power sequencer; verified with WAIT_TIME=1.

28.8.21 HDMI_NV_PDISP_SOR_SEQ_INST9_0

Usage: boot / initialization

Offset: 0x69 │ Byte Offset: 0x1a4 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.22 HDMI_NV_PDISP_SOR_SEQ_INSTA_0

Usage: boot / initialization

Offset: 0x6a │ Byte Offset: 0x1a8 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN

Bit	Reset	Description
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.23 HDMI_NV_PDISP_SOR_SEQ_INSTB_0

Usage: boot / initialization

Offset: 0x6b │ Byte Offset: 0x1ac │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV

Bit	Reset	Description
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.24 HDMI_NV_PDISP_SOR_SEQ_INSTC_0

Usage: boot / initialization

Offset: 0x6c │ Byte Offset: 0x1b0 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE

Bit	Reset	Description
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.25 HDMI_NV_PDISP_SOR_SEQ_INSTD_0

Usage: boot / initialization

Offset: 0x6d │ Byte Offset: 0x1b4 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE

Bit	Reset	Description
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.26 HDMI_NV_PDISP_SOR_SEQ_INSTE_0

Usage: boot / initialization

Offset: 0x6e │ Byte Offset: 0x1b8 │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE

Bit	Reset	Description
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.27 HDMI_NV_PDISP_SOR_SEQ_INSTF_0

Usage: boot / initialization

Offset: 0x6f │ Byte Offset: 0x1bc │ Read/Write: R/W │ Reset: 0x01008000 (0b00000001000xxxxx1x00xx0000000000)

Bit	Reset	Description
31	0x0	PLL_PULLDOWN: 0 = DISABLE 1 = ENABLE
30	0x0	POWERDOWN_MACRO: 0 = NORMAL 1 = POWERDOWN
29	0x0	ASSERT_PLL_RESETV: 0 = NORMAL 1 = RESETV
28	0x0	BLANK_V: 0 = NORMAL 1 = INACTIVE
27	0x0	BLANK_H: 0 = NORMAL 1 = INACTIVE
26	0x0	BLANK_DE: 0 = NORMAL 1 = INACTIVE
25	0x0	BLACK_DATA: 0 = NORMAL 1 = BLACK

Bit	Reset	Description
24	0x1	TRISTATE_IOS: 0 = ENABLE_PINS 1 = TRISTATE
23	0x0	DRIVE_PWM_OUT_LO: 0 = FALSE 1 = TRUE
22	0x0	PIN_B: 0 = LOW 1 = HIGH
21	0x0	PIN_A: 0 = LOW 1 = HIGH
15	0x1	HALT: 0 = FALSE 1 = TRUE
13:12	0x0	WAIT_UNITS: 0 = US 1 = MS 2 = VSYNC
9:0	0x0	WAIT_TIME

28.8.28 HDMI_NV_PDISP_SOR_LANE_DRIVE_CURRENT_0

TMDS per-lane I/O current control.

Although each control is specified as 8 bits below, making it easy to set the value in hex, only the 7 LSBs of each lane are used. The mapping of binary values to current is as follows:

- 0000000: 0 mA
- 0000001: 0.4mA
- 0000010: 0.8mA
- 0000011: 1.2mA
- 0000100: 1.6mA
- 0000101: 2.0mA
- 0000110: 2.4mA
- 0000111: 2.8mA
- 0001000: 3.2mA
- 0001001: 3.6mA
- 0001010: 4.0mA
- 0001011: 4.4mA
- 0001100: 4.8mA
- 0001101: 5.2mA
- 0001110: 5.6mA
- 0001111: 6.0mA
- 0010000: 6.4mA
- 0010001: 6.8mA
- 0010010: 7.2mA
- 0010011: 7.6mA

- 0010100: 8.0mA
- 0010101: 8.4mA
- 0010110: 8.8mA
- 0010111: 9.2mA
- 0011000: 9.6mA
- 0011001: 10.0mA
- 0011010: 10.4mA
- 0011011: 10.8mA
- 0011100: 11.2mA
- 0011101: 11.6mA
- 0011110: 12.0mA
- 0011111: 12.4mA
- 0100000: 12.8mA
- 0100001: 13.2mA
- 0100010: 13.6mA
- 0100011: 14.0mA
- 0100100: 14.4mA
- 0100101: 14.8mA
- 0100110: 15.2mA
- 0100111: 15.6mA
- 0101000: 16.0mA
- 0101001: 16.4mA
- 0101010: 16.8mA
- 0101011: 17.2mA
- 0101100: 17.6mA
- 0101101: 18.0mA
- 0101110: 18.4mA
- 0101111: 18.8mA
- 0110000: 19.2mA (start to borrow pre-amp current)
- 0110001: 19.6mA
- 0110010: 20.0mA
- 0110011: 20.4mA
- 0110100: 20.8mA
- 0110101: 21.2mA
- 0110110: 21.6mA
- 0110111: 22.0mA
- 0111000: 22.4mA
- 0111001: 22.8mA
- 0111010: 23.2mA
- 0111011: 23.6mA
- 0111100: 24.0 mA

- 0111101: 24.4mA
- 0111110: 24.8mA
- 0111111: 25.2mA
- 1000000: 25.4mA
- 1000001: 25.8mA
- 1000010: 26.2mA
- 1000011: 26.6mA
- 1000100: 27.0mA
- 1000101: 27.4mA
- 1000110: 27.8mA
- 1000111: 28.2mA (maximum value)
- 1001000: 25.4mA
- 1001001: 25.8mA
- 1001010: 26.2mA
- 1001011: 26.6mA
- 1001100: 27.0mA
- 1001101: 27.4mA
- 1001110: 27.8mA
- 1001111: 28.2mA
- 1010000: 25.4mA
- ...

1111111: 28.2 mA

By default, FUSE_OVERRIDE is FALSE, meaning that the TMDS current control is driven via calibration data stored in the fuse block. Software can override these defaults by programming FUSE_OVERRIDE to TRUE, and setting each LANEn field to the appropriate value.

Offset: 0x7e │ Byte Offset: 0x1f8 │ Read/Write: R/W │ Reset: 0x20202020 (0b00100000001000000010000000100000)

Bit	Reset	Description
31:24	0x20	LANE3
23:16	0x20	LANE2
15:8	0x20	LANE1
7:0	0x20	LANE0

28.8.29 HDMI_NV_PDISP_SOR_REFCLK_0

SOR_REFCLK

The HDMI clock is programmable and varies, depending on screen resolution. The NV_PDISP_SOR_SEQ_INSTn instructions (above) can wait for certain time intervals to elapse. Whenever hdmi_clk frequency is changed, this register must be reprogrammed to divide hdmi_clk to produce a 1 µs time reference, otherwise the time intervals requested by NV_PDISP_SOR_SEQ_INSTn will not be accurate.

The format of DIVISOR is an unsigned 8.2 divider. Because this is a simple digital divider, not a PLL, fractional values result in a jitter of one hdmi_clk between successive "1 µs" intervals, but the long-term average works out to the requested divisor. This

jitter is OK because the NV_PDISP_SOR_SEQ_INST wait intervals do not need to be exact. If the integer part is written as 0, it will be interpreted the same as "1".

Offset: 0x95 │ Byte Offset: 0x254 │ Read/Write: R/W │ Reset: 0x00001900 (0bxxxxxxxxxxxxxxxx0001100100xxxxxx)

Bit	Reset	Description
15:8	0x19	DIV_INT: default: 27 MHz
7:6	0x0	DIV_FRAC

28.8.30 HDMI_NV_PDISP_SOR_IO_PEAK_CURRENT_0

Pad controls for 28nm macro TMDS_X4_HP, 8 bits per lane

Transmitter De-emphasis Current

The register fields are 8 bits wide but only the 7 LSBs are currently used. The mapping of register value to current is as follows:

- 0000000: 0mA
- 0000001: 0.2mA
- 0000010: 0.4mA
- 0000011: 0.6mA
- 0000100: 0.8mA
- 0000101: 1.0mA
- 0000110: 1.2mA
- 0000111: 1.4mA
- 0001000: 1.6mA
- 0001001: 1.8mA
- 0001010: 2.0mA
- 0001011: 2.2mA
- 0001100: 2.4mA
- 0001101: 2.6mA
- 0001110: 2.8mA
- 0001111: 3.0mA
- 0010000: 3.2mA
- 0010001: 3.4mA
- 0010010: 3.6mA
- 0010011: 3.8mA
- 0010100: 4.0mA
- 0010101: 4.2mA
- 0010110: 4.4mA
- 0010111: 4.6mA
- 0011000: 4.8mA
- 0011001: 5.0mA
- 0011010: 5.2mA
- 0011011: 5.4mA

- 0011100: 5.6mA
- 0011101: 5.8mA
- 0011110: 6.0mA
- 0011111: 6.2mA
- 0100000: 6.4mA
- 0100001: 6.6mA
- 0100010: 6.8mA
- 0100011: 7.0mA
- 0100100: 7.2mA
- 0100101: 7.4mA
- 0100110: 7.6mA
- 0100111: 7.8mA
- 0101000: 8.0mA
- 0101001: 8.2mA
- 0101010: 8.4mA
- 0101011: 8.6mA
- 0101100: 8.8mA
- 0101101: 9.0mA
- 0101110: 9.2mA
- 0101111: 9.4mA (maximum pre-amp current)
- 0110000: 8.0mA
- 0110001: 8.2mA
- 0110010: 8.4mA
- 0110011: 8.6mA
- 0110100: 8.8mA
- 0110101: 9.0mA
- 0110110: 9.2mA
- 0110111: 9.4mA
- 0111000: 8.0mA
- ...
- 1111111: 9.4mA

Offset: 0xd1 │ Byte Offset: 0x344 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	LANE3
23:16	0x0	LANE2
15:8	0x0	LANE1
7:0	0x0	LANE0

28.8.31 HDMI_NV_PDISP_SOR_PAD_CTLS0_0

Offset: 0xd2 | Byte Offset: 0x348 | Read/Write: R/W | Reset: 0x000034bb (0b0xxxxx00x00000000011010010111011) | Default: 0x80000000

Bit	Reset	Default	Description
31	FALSE	TRUE	FUSE_OVERRIDE: 0 = FALSE 1 = TRUE (software default)
25	0x0	_NONE_	LOADADJ_SYNC_EN
24	0x0	_NONE_	LOADADJ_BYPN
22	0x0	_NONE_	REG_BYPASS
21	0x0	_NONE_	KVCO_2NDVCO
20	0x0	_NONE_	VCOCALIB_TS0
19	0x0	_NONE_	VCOCALIB_OVRWRB
18	0x0	_NONE_	VCOCALIB_ENB
17	0x0	_NONE_	VCOLIMIT_DISABLE
16	0x0	_NONE_	VCOLIMIT_SEL
15:12	0x3	_NONE_	BG_TEMP_COEF
11:8	0x4	_NONE_	BG_VREF_LEVEL
7:6	0x2	_NONE_	AVDD23_LEVEL
5:4	0x3	_NONE_	AVDD23_LOAD
3:2	0x2	_NONE_	AVDD10_LEVEL
1:0	0x3	_NONE_	AVDD10_LOAD

28.8.32 HDMI_NV_PDISP_SOR_PAD_CTLS1_0

Offset: 0xd3 | Byte Offset: 0x34c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11	0x0	DIV_RATIO_OVERRIDE
10	0x0	PLL_BYPASS
9:8	0x0	KICKSTART
7:4	0x0	PLL_NDIV_RATIO
3:0	0x0	PLL_PDIV_RATIO

28.9 Test and Debug Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

28.9.1 HDMI_NV_PDISP_SOR_VCRCA0_0

To quickly determine if the TMDS is working properly, a running CRC stamp is kept, which is reset at each VSYNC. For each sub-link, there are up to 40 bits of encoded data that have been re-parallelized from the serial data going out.

These encoded bits of data are broken up into three 16-bit pieces depending on either LVDS or TMDS encoding:

For TMDS, the encoded data is as follows:

- CRCH: {8'b00000000,TMDS_CLK_ENC[9:2]}
- CRCM: {TMDS_CLK_ENC[1:0],TMDS_TX2_ENC[9:0],TMDS_TX1_ENC[9:6]}
- CRCL: {TMDS_TX1_ENC[5:0],TMDS_TX0_ENC[9:0]}

For LVDS, the encoded data is as follows:

- CRCH: {8'b00000000,LVDS_CLK_ENC[6:4]}
- CRCM: {LVDS_CLK_ENC[3:0],,LVDS_A3_ENC[6:0],LVDS_A2_ENC[6:2]}
- CRCL: {LVDS_A2_ENC[1:0],LVDS_A1_ENC[6:0],LVDS_A0_ENC[6:0]}

For each of these pieces, a CRC16 is computed, CRCH for the upper 16-bit piece, CRCM forms the middle 16-bit piece, and CRCL for the lower 16-bit piece. The CRC16 takes in the 16 bits of encoded data plus the CRC16 value of the previous cycle to generate the new CRC16. The CRC equation which is used is:

x^16 + x^11 + x^4 + 1.

At each VSync, the previously running CRC16 values are captured into the VCRC registers and a previous CRC value of all zeroes is fed into the CRC16 units. This means the running CRC16 is reset at each VSync.

Note: For power saving, the VCRC is only computed when NV_PDISP_SOR_TRIG != 0. Also, it seems these are not latched at VSync but are free-running.

The VCRC registers for sub-link A

Usage: debug

Offset: 0x72 | Byte Offset: 0x1c8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	CRCM
15:0	X	CRCL

28.9.2 HDMI_NV_PDISP_SOR_VCRCA1_0

Usage: debug

Offset: 0x73 | Byte Offset: 0x1cc | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	CRCH

28.9.3 HDMI_NV_PDISP_SOR_CCRCA0_0

The CCRC registers contain the CRC value for the encoded link, captured when the trigger event occurred.

The CCRC registers for sub-link A

Usage: debug

Offset: 0x74 | Byte Offset: 0x1d0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	CRCM
15:0	X	CRCL

28.9.4 HDMI_NV_PDISP_SOR_CCRCA1_0

Usage: debug

Offset: 0x75 | Byte Offset: 0x1d4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	CRCH

28.9.5 HDMI_NV_PDISP_SOR_EDATAA0_0

EDATA has the encoded data for the serial channels TXC, TX3, TX2, TX1, and TX0. For each link, there are up to 40 bits of encoded data. The encoded bits making up the data are as follows:

For TMDS, the encoded data is as follows:

```
EDATA[39:0] =
{TMDS_CLK_ENC[9:0],
TMDS_TX2_ENC[9:0],
TMDS_TX1_ENC[9:0],
TMDS_TX0_ENC[9:0]}
```

For LVDS, the encoded data is as follows:

```
EDATA[39:0] =
{5'b00000,
LVDS_CLK_ENC[6:0],
LVDS_A3_ENC[6:0],
LVDS_A2_ENC[6:0],
LVDS A1 ENC[6:0],
LVDS_A0_ENC[6:0]}
```

The EDATA registers can be written to when any of the trigger bits for capture are set high.

The EDATA registers for sub-link A

Usage: debug

Offset: 0x76 | Byte Offset: 0x1d8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VAL

28.9.6 HDMI_NV_PDISP_SOR_EDATAA1_0

Usage: debug

Offset: 0x77 | Byte Offset: 0x1dc | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VAL

28.9.7 HDMI_NV_PDISP_SOR_COUNTA0_0

There are three 16-bit counts for each sub-link (TX0, TX1, TX2). The count registers for sub-link A.

Usage: debug

Offset: 0x78 | Byte Offset: 0x1e0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TX1
15:0	X	TX0

28.9.8 HDMI_NV_PDISP_SOR_COUNTA1_0

Usage: debug

Offset: 0x79 | Byte Offset: 0x1e4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	TX2

28.9.9 HDMI_NV_PDISP_SOR_DEBUGA0_0

The debug registers for sub-link A

The following registers control the debug data input to the serializer. These bits are only relevant when TEST_ENABLE is set.

DEBUGA0 has the data for the serial channels TXC, TX3, TX2, TX1, and TX0. For each link, there are up to 40 bits of data to be encoded. The bits to be encoded which make up the data are as follows:

For TMDS, the data to be encoded are loaded as follows:

```
{TMDS_CLK_ENC[9:0], TMDS_TX2_ENC[9:0], TMDS_TX1_ENC[9:0], TMDS_TX0_ENC[9:0]}
 =
 {DEBUGA1, DEBUGA0}
```

For LVDS, the encoded data is as follows:

```
 {5'b00000, LVDS_CLK_ENC[6:0], LVDS_A3_ENC[6:0], LVDS_A2_ENC[6:0],
LVDS_A1_ENC[6:0], LVDS_A0_ENC[6:0]}
 =
 {DEBUGA1[2:0], DEBUGA0}
```

Usage: debug

Offset: 0x7a | Byte Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	VAL: 0 = RESETV

28.9.10 HDMI_NV_PDISP_SOR_DEBUGA1_0

Usage: debug

Offset: 0x7b │ Byte Offset: 0x1ec │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	VAL: 0 = RESETV

28.9.11 HDMI_NV_PDISP_SOR_TRIG_0

TRIG specifies the number of pixel clock cycles after the previous VSYNC was detected to capture state data into IDATA, EDATA, CNT, and CCRC. After the trigger has captured data, the trigger gets reset with the next VSYNC and waits TRIG pixel cycles to capture data again. If the TRIG value is greater than the number of cycles between consecutive VSYNCs, then the trigger is not reset while it is still pending to capture data. Once the data has been captured, the trigger is reset with the following VSYNC to capture data again. If TRIG is all zeros, then the trigger is disabled and no capturing occurs.

Usage: debug

Offset: 0x7c │ Byte Offset: 0x1f0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:0	0x0	VAL: 0 = RESETV

28.9.12 HDMI_NV_PDISP_SOR_MSCHECK_0

The mode switch monitor is used for hardware debug only. To use, the test should set the CTL bit to CLEAR and then to RUN. At the end of the test, the register can be read. The various fields indicate the number of times the following conditions occurred:

- CRC enable went from false to true
- CRC enable went from true to false
- Data enable went from false to true
- Data enable went from true to false

All counts have 4 bits, so the count will clamp at 15.

Usage: debug

Offset: 0x7d │ Byte Offset: 0x1f4 │ Read/Write: R/W │ Reset: 0x80000000 (0b1xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	0x1	CTL: 0 = CLEAR 1 = RUN
15:12	RO	X	DATA_ENABLE_T2F
11:8	RO	X	DATA_ENABLE_F2T
7:4	RO	X	CRC_ENABLE_T2F
3:0	RO	X	CRC_ENABLE_F2T

28.10 Audio Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

28.10.1 HDMI_NV_PDISP_AUDIO_DEBUG0_0

This is the register space for the S/PDIF audio decoding block. Please refer to IEC60958-1 for general and IEC60958-3 for consumer specifications.

AUDIO_DEBUG0

This register is meant for debugging. It indicates various possible errors in the incoming S/PDIF stream or FIFO overflows. _NO indicates that no error of this type has occurred. _YES indicates that an error of the given type occurred at least once.

This is a write-1-to-clear register. To clear the _ERROR bits, just write _RESET to the bit you want to clear.

THRESHOLD_ERROR: When THRESHOLD_LOW is bigger than THRESHOLD_HIGH, this bit will be set. This indicates the internal state machine has reached an invalid state likely due to glitches in the input or the input being disconnected/reconnected.

HA/HB_FIFO_ERROR: The data is passed from the audio to the HDMI block via an async FIFO. This bit indicates that the FIFO is full. There is one bit for each head. This is an indication that the HDMI unit does not drain the audio sample data fast enough. We should only be looking at the bit for the head currently running HDMI. The bit for the other head will always be set since that FIFO will never be drained.

FRAME_ERROR: A block should have 192 frames. When a non-192 frames' block is detected, this bit will be set. Again, this indicates a problem with the incoming S/PDIF stream.

BITS_ERROR: A frame should have a 56-bit audio sample + an 8-bit preamble. If a non 56-bit audio sample is detected in a frame, this bit will be set.

OVERFLOW_ERROR: If the S/PDIF input stream is floating, the 10-bit pulse width counter will be overflowed, and this bit will be set. See OVERFLOW_TH for detail.

PREAMBLE_ERROR: A frame should have a 56-bit audio sample + an 8-bit preamble. If there is no valid preamble for a given sample, this bit will be set.

Offset: 0x7f │ Byte Offset: 0x1fc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxx0xxx0xxx0xxx0xxx0xxx0xxx0)

Bit	Reset	Description
24	0x0	PREAMBLE_ERROR: 0 = NO 1 = YES 1 = RESETV
20	0x0	OVERFLOW_ERROR: S/PDIF input stream is floating for Tegra 4 devices, so ignore the related overflow error. 0 = NO 1 = YES 1 = RESETV
16	0x0	BITS_ERROR: 0 = NO 1 = YES 1 = RESETV
12	0x0	FRAME_ERROR: 0 = NO 1 = YES 1 = RESETV

Bit	Reset	Description
8	0x0	HB_FIFO_ERROR: 0 = NO 1 = YES 1 = RESETV
4	0x0	HA_FIFO_ERROR: 0 = NO 1 = YES 1 = RESETV
0	0x0	THRESHOLD_ERROR: 0 = NO 1 = YES 1 = RESETV

28.10.2 HDMI_NV_PDISP_AUDIO_DEBUG1_0

AUDIO_DEBUG1

OVERFLOW_TH (bit 9:0): There is a 10-bit counter running on dispclk_audio_fs and counting S/PDIF's input pulse length. The maximum pulse length count is 305, when dispclk=416 MHz and fs=32 kHz. If someone unplugged the S/PDIF input connector, this counter will count above 305 (0x131) and then overflow eventually. OVERFLOW_TH is the threshold that being used to conclude when the S/PDIF input is being unplugged or floating. If the 10-bit counter is greater than or equal to OVERFLOW_TH, then OVERFLOW_ERROR bit will be set.

Software should not need to program this register to anything but its initialized value. The initialized value is the maximum value that the 10-bit counter can reach before overflowing.

PACKET_NOISE_FILTER_ENABLE: A register delays packets long enough to determine if the packet was of a legitimate length. If a packet contains more than 56 bits, the packet is dropped and an error condition is flagged (NV_PDISP_AUDIO_DEBUG0_PREAMBLE_ERROR = _YES). This register is included as a debug to allow disabling of the packet delay and dropping functionality.

When ENABLE is _YES, bad audio packets will be dropped and preamble errors flagged

When ENABLE is _NO, bad audio frames will be sent through as is.

PREAMBLE_RECOVER_DEBUG: When YES, audio recover circuit accepts a new preamble and audio sample even when it arrives unexpectedly (i.e., before previous sample has completed). NO restores legacy behavior, which is to drop the new sample. Software should not need to change this bit.

Offset: 0x80 │ Byte Offset: 0x200 │ Read/Write: R/W │ Reset: 0x800103ff (0b1xxxxxxxxxxxxxx1xxxxxx1111111111)

Bit	Reset	Description
31	YES	PREAMBLE_RECOVER_DEBUG: 0 = NO 1 = YES
16	0x1	PACKET_NOISE_FILTER_ENABLE: 0 = NO 1 = YES
9:0	0x3ff	OVERFLOW_TH

28.10.3 HDMI_NV_PDISP_AUDIO_DEBUG2_0

AUDIO_DEBUG2

There is an LFSR (linear feedback shift register) that can generate pseudo- random data to feed the output FIFO rather than use the incoming SPDIF stream. This is useful for testing in emulation or bring-up; a feature that software should not worry about.

LFSR_COUNT: LFSR_COUNT is the rate of filling LFSR data into the FIFO. The rate needs to be close to actual S/PDIF data input rate, which is depended on fs (audio sampling frequency). Set LFSR_COUNT according to the following table:

For dispclk_audio_fs = 416 MHz:

- fs = 32.0KHz, LFSR_COUNT = 0x32DC
- fs = 44.1KHz, LFSR_COUNT = 0x24E8
- fs = 48.0KHz, LFSR_COUNT = 0x21E8
- fs = 88.2KHz, LFSR_COUNT = 0x1274
- fs = 96.0KHz, LFSR_COUNT = 0x10F4
- fs = 176.4KHz, LFSR_COUNT = 0x093A
- fs = 192.0KHz, LFSR_COUNT = 0x087A

For dispclk_audio_fs = 277 MHz:

- fs = 32.0KHz, LFSR_COUNT = 0x21EB
- fs = 44.1KHz, LFSR_COUNT = 0x189A
- fs = 48.0KHz, LFSR_COUNT = 0x169B
- fs = 88.2KHz, LFSR_COUNT = 0x0C4D
- fs = 96.0KHz, LFSR_COUNT = 0x0B4D
- fs = 176.4KHz, LFSR_COUNT = 0x0626
- fs = 192.0KHz, LFSR_COUNT = 0x05A6

CHANNEL_STATUS: This bit does not do anything right now, since the channel status bit is generated on HDMI side.

USER: This is the user bit that is going into the FIFO when LFSR_DATA_EN is set.

VALIDITY: This is the validity bit that is going into the FIFO when LFSR_DATA_EN is set.

LFSR_DATA_EN: When this bit is set, the FIFO will be filled with LFSR (Linear Feedback Shift Register) data and ignore incoming S/PDIF input stream. LFSR will generate pseudo-random data in following pattern:

F->E->C->8->1->2->4->9->3->6->D->A->5->B->7->F->E.....

Offset: 0x81 │ Byte Offset: 0x204 │ Read/Write: R/W │ Reset: 0x000021e8 (0b0xxxxxx0xxx0xxx00010000111101000)

Bit	Reset	Description
31	0x0	LFSR_DATA_EN: 0 = NO 1 = YES
24	0x0	VALIDITY
20	0x0	USER
16	0x0	CHANNEL_STATUS
15:0	0x21e8	LFSR_COUNT

28.10.4 HDMI_NV_PDISP_AUDIO_FS1_0

When a B preamble (a B preamble is an indicator of the start of a frame in S/PDIF) is detected, a counter that is running on dispclk_audio_fs will count the number of clocks needed for the 8 HALFs (or 4 bits) B preamble length and compare against each of FSx_LOW and FSx_HIGH. (where x goes from 1 to 7, one entry per audio frequency).

For one of the 7 cases, we will have the case that:

FSx_LOW <= 8 HALF count <= FSx_HIGH

and the hardware will report that the incoming stream is running at frequency 'x' in the AUDIO_CNTRL0_SAMPLING_FREQ register field.

- FS1 is for fs = 32.0 KHz.
- FS2 is for fs = 44.1 KHz.
- FS3 is for fs = 48.0 KHz.
- FS4 is for fs = 88.2 KHz.
- FS5 is for fs = 96.0 KHz.
- FS6 is for fs = 176.4 KHz.
- FS7 is for fs = 192.0 KHz.

The formula to compute 8 HALFs is:

8 HALFs = 8 * hdmi_audio_clk / (fs * 128)

The value programmed in FSx_HIGH should be 8 HALF count + 10

The value programmed in FSx_LOW should be 8 HALF count - 10

For example: if the audio sample frequency is 32 kHz and dispclk_audio_fs is 216 MHz, then the 8 HALFs (or 4 bits) preamble length is:

8 HALFs = 8 * 216.0 MHz / (32.0 kHz * 128)

= 421 (or 422)

So, when FS1_LOW = 412 (0x19c) and FS1_HIGH = 430 (0x1ae), the 412 <= 8 HALFs <= 430 condition will be true. And AUDIO_CNTRL0_SAMPLING_FREQ will be equal to _32_0KHZ, which means 32 kHz of audio sampling frequency is detected.

The init values of the registers below are based on an hdmi_audio_clk of 216 MHz, and software should reprogram these every time that changes (which should not happen).

AUDIO_FS1

LOW: Low threshold (in dispclk periods) for 32 kHz audio detection.

HIGH: High threshold (in dispclk periods) for 32 kHz audio detection.

Offset: 0x82 │ Byte Offset: 0x208 │ Read/Write: R/W │ Reset: 0x01ae019c (0bxxxx000110101110xxxx000110011100)

Bit	Reset	Description
27:16	0x1ae	HIGH
11:0	0x19c	LOW

28.10.5 HDMI_NV_PDISP_AUDIO_FS2_0

AUDIO_FS2

LOW: Low threshold (in dispclk periods) for 44.1 kHz audio detection.

HIGH: High threshold (in dispclk periods) for 44.1 kHz audio detection.

Offset: 0x83 | Byte Offset: 0x20c | Read/Write: R/W | Reset: 0x013b0129 (0bxxxx000100111011xxxx000100101001)

Bit	Reset	Description
27:16	0x13b	HIGH
11:0	0x129	LOW

28.10.6 HDMI_NV_PDISP_AUDIO_FS3_0

AUDIO_FS3

LOW: Low threshold (in dispclk periods) for 48 kHz audio detection.

HIGH: High threshold (in dispclk periods) for 48 kHz audio detection.

Offset: 0x84 | Byte Offset: 0x210 | Read/Write: R/W | Reset: 0x01220110 (0bxxxx000100100010xxxx000100010000)

Bit	Reset	Description
27:16	0x122	HIGH
11:0	0x110	LOW

28.10.7 HDMI_NV_PDISP_AUDIO_FS4_0

AUDIO_FS4

LOW: Low threshold (in dispclk periods) for 88.2 kHz audio detection.

HIGH: High threshold (in dispclk periods) for 88.2 kHz audio detection.

Offset: 0x85 | Byte Offset: 0x214 | Read/Write: R/W | Reset: 0x009f0093 (0bxxxx000010011111xxxx000010010011)

Bit	Reset	Description
27:16	0x9f	HIGH
11:0	0x93	LOW

28.10.8 HDMI_NV_PDISP_AUDIO_FS5_0

AUDIO_FS5

LOW: Low threshold (in dispclk periods) for 96 kHz audio detection.

HIGH: High threshold (in dispclk periods) for 96 kHz audio detection.

Offset: 0x86 | Byte Offset: 0x218 | Read/Write: R/W | Reset: 0x00920086 (0bxxxx000010010010xxxx000010000110)

Bit	Reset	Description
27:16	0x92	HIGH
11:0	0x86	LOW

28.10.9 HDMI_NV_PDISP_AUDIO_FS6_0

AUDIO_FS6

LOW: Low threshold (in dispclk periods) for 176.4 kHz audio detection.

HIGH: High threshold (in dispclk periods) for 176.4 kHz audio detection.

Offset: 0x87 │ Byte Offset: 0x21c │ Read/Write: R/W │ Reset: 0x004e004a (0bxxxx000001001110xxxx000001001010)

Bit	Reset	Description
27:16	0x4e	HIGH
11:0	0x4a	LOW

28.10.10 HDMI_NV_PDISP_AUDIO_FS7_0

AUDIO_FS7

LOW: Low threshold (in dispclk periods) for 192 kHz audio detection.

HIGH: High threshold (in dispclk periods) for 192 kHz audio detection.

Offset: 0x88 │ Byte Offset: 0x220 │ Read/Write: R/W │ Reset: 0x00480044 (0bxxxx000001001000xxxx000001000100)

Bit	Reset	Description
27:16	0x48	HIGH
11:0	0x44	LOW

28.10.11 HDMI_NV_PDISP_AUDIO_PULSE_WIDTH_0

AUDIO_PULSE_WIDTH

This is a status only register used for debugging.

HALF: Because the S/PDIF input stream is in biphase coding, a 32 bit sub-frame is broken down to 64 states. HALF (half bit or 1 state) is the number of dispclk_audio_fs periods for a minimum pulse length that can be measured on the S/PDIF input stream at any given audio fs.

For example, HALF = dispclk_audio_fs / (fs * 128)

THREE_HALF: THREE_HALF is the number of dispclk_audio_fs periods for the maximum pulse length in the S/PDIF input stream. THREE_HALF should be approximately 3 times of HALF; this only happens in the B, M, or W preamble.

COUNT2LOCK: If LOCK is set, COUNT2LOCK is number of frames to skip before locking down the HALF and THREE_HALF values. This field has no effect if LOCK is not set.

LOCK: If LOCK is set, it locks down the HALF and THREE_HALF values. If LOCK is clear, then HALF and THREE_HALF are updated every frame (B or M preamble) Setting LOCK means that the hardware will not detect changes in speeds of the incoming audio stream.

This mode should not be used by software. It is meant as a debugging tool.

Offset: 0x89 │ Byte Offset: 0x224 │ Read/Write: R/W │ Reset: 0x01XXXXXX (0bxxx0xx01xxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
28	RW	0x0	LOCK: 0 = NO 1 = YES

Bit	R/W	Reset	Description
25:24	RW	0x1	COUNT2LOCK
21:12	RO	X	THREE_HALF
9:0	RO	X	HALF

28.10.12 HDMI_NV_PDISP_AUDIO_THRESHOLD_0

AUDIO_THRESHOLD

This status-only register is useful for debugging. It provides information about some characteristics of the S/PDIF stream.

THRESHOLD_LOW: THRESHOLD_LOW is 1.5 times of HALF. If a S/PDIF pulse is less than or equal to THRESHOLD_LOW, then a bit '1' is detected.

THRESHOLD_HIGH: THRESHOLD_HIGH is 2.5 times of HALF. If a S/PDIF pulse is less than or equal to THRESHOLD_HIGH and bigger than THRESHOLD_LOW, then a bit '0' is detected.

Offset: 0x8a │ Byte Offset: 0x228 │ Read/Write: RO │ Reset: 0x00XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
21:12	X	HIGH
9:0	X	LOW

28.10.13 HDMI_NV_PDISP_AUDIO_CNTRL0_0

AUDIO_CNTRL0

ERROR_TOLERANCE: Because the (dispclk_audio_fs / audio sampling frequency) is a non-integer in general and due to the S/PDIF input stream's jitter, THREE_HALF is not exactly equal to 3 times of HALF. ERROR_TOLERANCE is the error count in dispclk periods allowed for 3 x HALF vs THREE_HALF. Based on simulation, the value of 4 to 7 is a good value when dispclk_audio_fs = 416 MHz. It is anticipated that software does not need to change this field from its init value.

SOFT_RESET: Reset the frame_counter and wait for the next B preamble to resume the audio data transfer. This is intended to clear registers in case the hardware ends up in a bad state, and should only be used for debug.

SOFT_RESET_ALL: Reset all register in audio block. This is currently a placeholder and is not implemented.

SAMPLING_FREQ: This will report the incoming S/PDIF audio stream sampling frequency. The HDMI specification only supports 7 audio sample frequency (fs), anything other than those 7 fs, will be reported as UNKNOWN.

SOURCE_SELECT: Determines whether to use the S/PDIF or Azalia (HDAL) audio input. This field has no effect on chips without the HDAL input. The default is AUTO: selects S/PDIF audio until the HDAL audio input is initialized by the external controller.

FRAMES_PER_BLOCK: FRAMES_PER_BLOCK is number of frames per block. By specification, each block has 192 (0xC0) frames.

Offset: 0x8b │ Byte Offset: 0x22c │ Read/Write: R/W │ Reset: 0xc00X0006 (0b11000000xxxxxxxxxxx0xxx000000110)

Bit	R/W	Reset	Description
31:24	RW	0xc0	FRAMES_PER_BLOCK
19:16	RO	X	SAMPLING_FREQ: 1 = UNKNOWN 3 = FREQ_32KHZ 0 = FREQ_44_1KHZ 2 = FREQ_48KHZ

Bit	R/W	Reset	Description
			8 = FREQ_88_2KHZ 10 = FREQ_96KHZ 12 = FREQ_176_4KHZ 14 = FREQ_192KHZ
12	RW	0x0	SOFT_RESET_ALL: 1 = ALL_ASSERT 0 = ALL_DEASSERT
8	RW	0x0	SOFT_RESET: 1 = ASSERT 0 = DEASSERT
7:0	RW	0x6	ERROR_TOLERANCE

28.10.14 HDMI_NV_PDISP_AUDIO_N_0

AUDIO_N

N_VALUE: N_VALUE is the N parameter in HDMI Audio Clock Regeneration Packet. This value should be written when hardware measured CTS is enabled. The correct N value can be read from tables 7-1, 7-2, and 7-3 in the HDMI specification.

N_RESET: N_RESET is the reset for the N counter. If software is controlling the value of N, every time the audio stream changes sampling frequency (fs), software (driver) need to reset the N counter by writing _ASSERT to N_RESET followed by writing a _DEASSERT to N_RESET.

If the hardware selected N feature is enabled (N_LOOKUP = _ENABLE), software only needs to reset the N counter when writing to the AUDIO_NVAL registers and when enabling/disabling N_LOOKUP.

1) Set N_RESET = _ASSERT

2) Modify N value registers

3) Set N_RESET = _DEASSERT

Modifying N_VALUE can be done in step 1.

N_GENERATE: N_GENERATE controls how the audio block generates the 128*fs/N pulse, which is related to CTS. The detail of CTS can be found in HDMI 1.1 specification, Chapter 7. This bit is strictly for debugging purposes only.

_NORMAL: This method increments the CTS counter a variable number of times based on the length of the S/PDIF pulse.

_ALTERNATE: This method attempts to recreate the 128*fs clock and increments the CTS counter at regular intervals based on HALF.

N_LOOKUP: When set to _ENABLE, the hardware will select the appropriate value of N to use from one of the AUDIO_NVAL registers. This selection is based on the audio sampling frequency detected by the audio block. This is not the sampling frequency reported in the channel status bits. N_RESET should be toggled when this feature is enabled. When set to _DISABLE, software must program the correct N value into AUDIO_N.

Offset: 0x8c │ Byte Offset: 0x230 │ Read/Write: R/W │ Reset: 0x01000000 (0bxxx0xxx1xxx00000000000000000000)

Bit	Reset	Description
28	0x0	LOOKUP: 1 = ENABLE 0 = DISABLE
24	0x1	GENERATE: 0 = NORMAL 1 = ALTERNATE

Bit	Reset	Description
20	0x0	RESETF: 1 = ASSERT 0 = DEASSERT
19:0	0x0	VALUE

28.10.15 HDMI_NV_PDISP_AUDIO_NVAL_0320_0

The following seven registers are used when Hardware N lookup is enabled (AUDIO_N_LOOKUP). These registers should be written with the correct N values for the current pixel clock frequency. These values can be found in tables 7-1, 7-2, and 7-3 of the HDMI specification. These registers are initialized to the value in the "other" row of these tables.

These registers may need to be reprogrammed when the pixel clock frequency changes. When changing the value of these registers, the N counter should be reset by toggling _RESET in AUDIO_N.

VALUE: The correct N value for 32 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x8d │ Byte Offset: 0x234 │ Read/Write: R/W │ Reset: 0x00001000 (0bxxxxxxxxxxxxx0000001000000000000)

Bit	Reset	Description
19:0	0x1000	VAL_0320_VALUE

28.10.16 HDMI_NV_PDISP_AUDIO_NVAL_0441_0

VALUE: The correct N value for 44.1 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x8e │ Byte Offset: 0x238 │ Read/Write: R/W │ Reset: 0x00001880 (0bxxxxxxxxxxxxx0000001100010000000)

Bit	Reset	Description
19:0	0x1880	VAL_0441_VALUE

28.10.17 HDMI_NV_PDISP_AUDIO_NVAL_0882_0

VALUE: The correct N value for 88.2 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x8f │ Byte Offset: 0x23c │ Read/Write: R/W │ Reset: 0x00003100 (0bxxxxxxxxxxxxx0000011000100000000)

Bit	Reset	Description
19:0	0x3100	VAL_0882_VALUE

28.10.18 HDMI_NV_PDISP_AUDIO_NVAL_1764_0

VALUE: The correct N value for 176.4 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x90 │ Byte Offset: 0x240 │ Read/Write: R/W │ Reset: 0x00006200 (0bxxxxxxxxxxxxx0000110001000000000)

Bit	Reset	Description
19:0	0x6200	VAL_1764_VALUE

28.10.19 HDMI_NV_PDISP_AUDIO_NVAL_0480_0

VALUE: The correct N value for 48 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x91 │ Byte Offset: 0x244 │ Read/Write: R/W │ Reset: 0x00001800 (0bxxxxxxxxxxxx00000001100000000000)

Bit	Reset	Description
19:0	0x1800	VAL_0480_VALUE

28.10.20 HDMI_NV_PDISP_AUDIO_NVAL_0960_0

VALUE: The correct N value for 96 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x92 │ Byte Offset: 0x248 │ Read/Write: R/W │ Reset: 0x00003000 (0bxxxxxxxxxxxx00000011000000000000)

Bit	Reset	Description
19:0	0x3000	VAL_0960_VALUE

28.10.21 HDMI_NV_PDISP_AUDIO_NVAL_1920_0

VALUE: The correct N value for 192 kHz audio at the current pixel clock frequency should be written here.

Offset: 0x93 │ Byte Offset: 0x24c │ Read/Write: R/W │ Reset: 0x00006000 (0bxxxxxxxxxxxx00000110000000000000)

Bit	Reset	Description
19:0	0x6000	VAL_1920_VALUE

28.10.22 HDMI_NV_PDISP_SOR_AUDIO_CNTRL0_0

HD Audio (also known as Azalia)

PORT_CONNECTIVITY: This controls the behavior of the Port Connectivity field of the Azalia Configuration Defaults Verb. This can be used to disable a codec that is associated with an SOR that does not connect to a physical port.

- ENABLE: Report 00 in the Port Connectivity field by default. This corresponds to "The Port Complex is connected to a jack"
- DISABLE: Report 01 in the Port Connectivity field by default. This corresponds to "No physical connection for Port" See table 100 and table 101 of the High Definition Audio Specification (Revision 1.0) for more information.

AFIFO_FLUSH: When the DP and HDMI logic are not in the middle of sending an audio packet, they will flush out any entries at the head of the AFIFO that is not tagged as the beginning of a sample. This ensures that the next new audio packet sent will begin on the correct channel. This is a diagnostic workaround bit to disable the flushing.

- ENABLE: Automatically fix the AFIFO if it gets out of alignment
- DISABLE: Do not throw out any AFIFO entries.

SAMPLING_FREQ: This will report the incoming audio stream sampling frequency in the Azalia codec. The HDMI specification only supports 7 audio sample frequencies (fs). Anything other than those 7 fs will be reported as UNKNOWN. See the HDMI 1.1 specification, Table 7-4 (page 77).

SOURCE_SELECT: Determines whether to use the S/PDIF or Azalia (HDAL) audio input. This field has no effect on chips without the HDAL input. The default is AUTO: selects S/PDIF audio until the HDAL audio input is initialized by the external controller

INJECT_NULLSMPL: When the bit is enabled, if the audio format is stereo LPCM as indicated by the stream format in the corresponding output converter widget, the codec inserts null samples into the audio FIFO for each Azalia frame in which it did

not receive any samples. This is done only for stereo LPCM and not for any other audio format. This bit should be disabled by default for backwards compatibility.

INPUT_MODE: This bit indicates what the audio data source is. HDA is Azalia data, S/PDIF is S/PDIF data.

SOR_AUDIO_CNTRL0

Offset: 0xac │ Byte Offset: 0x2b0 │ Read/Write: R/W │ Reset: 0xX01X1000 (0bxx0xxxxxxx01xxxxxxx1xxxxxxxxxxx0)

Bit	R/W	Reset	Description
31	RO	X	INPUT_MODE: 0 = HDA 1 = SPDIF
29	RW	DISABLE	INJECT_NULLSMPL: 0 = DISABLE 1 = ENABLE
21:20	RW	SPDIF	SOURCE_SELECT: 0 = AUTO 1 = SPDIF 2 = HDAL
19:16	RO	X	SAMPLING_FREQ: 3 = FREQ_32_0KHZ 0 = FREQ_44_1KHZ 8 = FREQ_88_2KHZ 12 = FREQ_176_4KHZ 2 = FREQ_48_0KHZ 10 = FREQ_96_0KHZ 14 = FREQ_192_0KHZ 1 = FREQ_UNKNOWN
12	RW	ENABLED	AFIFO_FLUSH: 0 = DISABLED 1 = ENABLED
0	RW	ENABLE	PORT_CONNECTIVITY: 0 = ENABLE 1 = DISABLE

28.10.23 HDMI_NV_PDISP_SOR_AUDIO_DEBUG_0

SOR_AUDIO_DEBUG

This register is used for debug purposes only.

FIFO_ERROR: The data is passed from the audio (Azalia or S/PDIF) to the SOR via an async FIFO. This bit indicates that the FIFO is full. There is one bit for each head. This is an indication that the SOR units do not drain away the audio sample data fast enough.

Offset: 0xad │ Byte Offset: 0x2b4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	NO	FIFO_ERROR: 0 = NO 1 = YES

28.10.24 HDMI_NV_PDISP_SOR_AUDIO_SPARE0_0

This register is a backup register.

SOR_AUDIO_SPARE0

Offset: 0xae │ Byte Offset: 0x2b8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	Reserved.

28.10.25 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0320_0

SOR_AUDIO_NVAL

The following seven registers are used for HDMI N values for Azalia. These registers should be written with the correct N values for the current pixel clock frequency. These values can be found in tables 7-1, 7-2, and 7-3 of the HDMI specification. These registers are initialized to the value in the "other" row of these tables.

For DisplayPort audio, a value of 2^15 (0x8000) is always used.

VALUE: The correct N value for 32 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xaf │ Byte Offset: 0x2bc │ Read/Write: R/W │ Reset: 0x00001000 (0bxxxxxxxxxxxxx0000001000000000000)

Bit	Reset	Description
19:0	0x1000	VALUE

28.10.26 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0441_0

VALUE: The correct N value for 44.1 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xb0 │ Byte Offset: 0x2c0 │ Read/Write: R/W │ Reset: 0x00001880 (0bxxxxxxxxxxxxx0000001100010000000)

Bit	Reset	Description
19:0	0x1880	VALUE

28.10.27 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0882_0

VALUE: The correct N value for 88.2 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xb1 │ Byte Offset: 0x2c4 │ Read/Write: R/W │ Reset: 0x00003100 (0bxxxxxxxxxxxxx0000011000100000000)

Bit	Reset	Description
19:0	0x3100	VALUE

28.10.28 HDMI_NV_PDISP_SOR_AUDIO_NVAL_1764_0

VALUE: The correct N value for 176.4 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xb2 │ Byte Offset: 0x2c8 │ Read/Write: R/W │ Reset: 0x00006200 (0bxxxxxxxxxxxxx0000110001000000000)

Bit	Reset	Description
19:0	0x6200	VALUE

28.10.29 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0480_0

VALUE: The correct N value for 48 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xb3 | Byte Offset: 0x2cc | Read/Write: R/W | Reset: 0x00001800 (0bxxxxxxxxxxxx00000001100000000000)

Bit	Reset	Description
19:0	0x1800	VALUE

28.10.30 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0960_0

VALUE: The correct N value for 96 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xb4 | Byte Offset: 0x2d0 | Read/Write: R/W | Reset: 0x00003000 (0bxxxxxxxxxxxx00000011000000000000)

Bit	Reset	Description
19:0	0x3000	VALUE

28.10.31 HDMI_NV_PDISP_SOR_AUDIO_NVAL_1920_0

VALUE: The correct N value for 192 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xb5 | Byte Offset: 0x2d4 | Read/Write: R/W | Reset: 0x00006000 (0bxxxxxxxxxxxx00000110000000000000)

Bit	Reset	Description
19:0	0x6000	VALUE

28.10.32 HDMI_NV_PDISP_SOR_AUDIO_HDA_SCRATCH0_0

SOR_AUDIO_HDA_SCRATCH0/1/2/3

This is a field to be used if it is determined at a later time that additional information needs to be sent from the display driver to the audio driver for support of any of the extended formats.

Offset: 0xb6 | Byte Offset: 0x2d8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

28.10.33 HDMI_NV_PDISP_SOR_AUDIO_HDA_SCRATCH1_0

Offset: 0xb7 | Byte Offset: 0x2dc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

28.10.34 HDMI_NV_PDISP_SOR_AUDIO_HDA_SCRATCH2_0

Offset: 0xb8 | Byte Offset: 0x2e0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

28.10.35 HDMI_NV_PDISP_SOR_AUDIO_HDA_SCRATCH3_0

Offset: 0xb9 | Byte Offset: 0x2e4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	RESERVED

28.10.36 HDMI_NV_PDISP_SOR_AUDIO_HDA_CODEC_SCRATCH0_0

These registers are set by the audio driver using vendor-defined verbs. They can be used to pass information from the audio driver to the resource manager if that functionality is ever needed.

When bit 31 changes, an interrupt can be generated (see INT_STATUS register)

Offset: 0xba | Byte Offset: 0x2e8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

28.10.37 HDMI_NV_PDISP_SOR_AUDIO_HDA_CODEC_SCRATCH1_0

Offset: 0xbb | Byte Offset: 0x2ec | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA

28.10.38 HDMI_NV_PDISP_SOR_AUDIO_HDA_ELD_BUFWR_0

Software Programming Model

In an integrated graphics chip, which includes an NVIDIA® Audio Controller, with support for an NVIDIA audio codec driver, the following is the background and the suggested programming model.

The audio software for the HDMI codec will need information about the audio capabilities of an attached HDMI sink device. This information is stored in the HDMI sink device's EDID. Typically, the EDID flows through a graphics adapter to graphics software, so the graphics adapter hardware will not have knowledge of the EDID contents.

To that end, a new mechanism is defined for passing the HDMI sink device's audio EDID information from the graphics software to the audio software. The data payload containing the audio information will be known as EDID-Like Data (or ELD) and will contain a subset of the HDMI sink devices EDID information.

The ELD information will be valid if the HDMI sink is attached and powered on and the ELD Valid bit is set. The Pin Widget that is associated with this HDMI widget will report if the device is attached and that the ELD memory is populated and valid by reporting Presence Detect of 1 and ELD Valid of 1 to a Pin Sense control command. As with the Presence Detect bit, the changes to the ELD Valid bit can also result in the generation of unsolicited responses.

Each codec implements a 96-byte ELD buffer that is written by the resource manager and read by the audio driver. Once the ELD buffer is written by the resource manager, the valid bit, NV_PDISP_SOR_AUDIO_HDA_PRESENSE_ELDV is set to _VALID to indicate that ELD contents have been initialized by the resource manager. On hot unplug events, NV_PDISP_SOR_AUDIO_HDA_PRESENSE_ELDV is set to _INVALID to erase ELD programming in the audio driver.

Offset: 0xbc │ Byte Offset: 0x2f0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	INDEX
7:0	0x0	DATABYTE

28.10.39 HDMI_NV_PDISP_SOR_AUDIO_HDA_PRESENSE_0

Reports the Hot Plug state and ELD state to the audio driver. Changes to this state can cause an unsolicited response.

ELDV: Indicates whether the data in the ELD buffer is valid and ready to read.

PD: Presence Detect. This should be set to _PRESENT by software upon hot plug and _NOT_PRESENT on unplug.

Offset: 0xbd │ Byte Offset: 0x2f4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	INVALID	ELDV: 0 = INVALID 1 = VALID
0	NOT_PRESENT	PD: 0 = NOT_PRESENT 1 = PRESENT

28.10.40 HDMI_NV_PDISP_SOR_AUDIO_HDA_CP_0

Reports the Content Protection state requested by the Audio driver. It is at the discretion of the video driver to enable or disable content protection. This is a read-only register set by the Content Protection Control (CP_CONTROL) verb.

REQUEST_STATE

_DONT_CARE: No state change requested

_RESERVED: Unused

_PROTECTION_OFF: Audio driver requests content protection to be disabled.

_PROTECTION_ON: Audio driver requests content protection to be enabled.

Offset: 0xbe │ Byte Offset: 0x2f8 │ Read/Write: RO │ Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
2	X	REQUEST_STATE_VALID
1:0	X	REQUEST_STATE: 0 = DONT_CARE 2 = PROTECTION_OFF 3 = PROTECTION_ON

28.10.41 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0320_0

When using the Azalia codec, it is difficult to recover the 128*fs clock frequency from the incoming audio stream. Therefore, when using the Azalia codec, these registers must be programmed to emulate the N counter.

When using S/PDIF, the N counter will run at 128*fs (audio sampling frequency) and count to a value determined by the HDMI specification. The Azalia counter will run at 24 MHz at all times, and it needs to count for the same period of time as the N counter would have.

Refer to section 7.2 of the HDMI specification, 1.2.

AVAL values do not need to be programmed by software. Their default values are OK. If the NVAL changes, it will change the N counter frequency. AVAL will need to be changed to match with this new NVAL. Certain resolutions may require different NVALs.

For Mobile chips, the clock is 24 MHz, not 54 MHz. For 44.1, the default N of 6272 gives non-integer AVAL. Choosing N of 4704 instead (nominal CTS 61875) gives an N frequency of 1200 Hz.

48k: 1ms 24000 azaclk cycles (0x5DC0)

44.1k: 1ms/1.2 20000 azaclk cycles (0x4E20)

But this is academic because CTS/N is hardcoded since the pixel clock / audio clock ratios are known and fixed.

At 27 MHz Pixel Clock			
	CTS	**N**	**AVAL**
44.1	22500	4704	20000
88.2	22500	9408	20000
176.4	22500	18816	20000
At 74.25 MHz Pixel Clock			
44.1	61875	4704	20000
88.2	61875	9408	20000
176.4	61875	18816	20000
At 148.5 MHz Pixel Clock:			
44.1	123750	4704	20000
88.2	123750	9408	20000
176.4	123750	18816	20000

VALUE: The correct A value for 32 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xbf │ Byte Offset: 0x2fc │ Read/Write: R/W │ Reset: 0x00005dc0 (0bxxxxxxxxxxxxx0000010111011100000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

28.10.42 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0441_0

VALUE: The correct A value for 44.1 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xc0 │ Byte Offset: 0x300 │ Read/Write: R/W │ Reset: 0x00004e20 (0bxxxxxxxxxxxx0000010011100010000)

Bit	Reset	Description
19:0	0x4e20	VALUE

28.10.43 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0882_0

VALUE: The correct A value for 88.2 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xc1 │ Byte Offset: 0x304 │ Read/Write: R/W │ Reset: 0x00004e20 (0bxxxxxxxxxxxx0000010011100010000)

Bit	Reset	Description
19:0	0x4e20	VALUE

28.10.44 HDMI_NV_PDISP_SOR_AUDIO_AVAL_1764_0

VALUE: The correct A value for 176.4 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xc2 | Byte Offset: 0x308 | Read/Write: R/W | Reset: 0x00004e20 (0bxxxxxxxxxxxxx00000100111000100000)

Bit	Reset	Description
19:0	0x4e20	VALUE

28.10.45 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0480_0

VALUE: The correct A value for 48 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xc3 | Byte Offset: 0x30c | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxxx00000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

28.10.46 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0960_0

VALUE: The correct A value for 96 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xc4 | Byte Offset: 0x310 | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxxx00000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

28.10.47 HDMI_NV_PDISP_SOR_AUDIO_AVAL_1920_0

VALUE: The correct A value for 192 kHz audio at the current pixel clock frequency should be written here.

Offset: 0xc5 | Byte Offset: 0x314 | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxxx00000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

28.10.48 HDMI_NV_PDISP_SOR_AUDIO_AVAL_DEFAULT_0

VALUE: Default A value if the Azalia codec sampling frequency does not match any of the above.

Offset: 0xc6 | Byte Offset: 0x318 | Read/Write: R/W | Reset: 0x00005dc0 (0bxxxxxxxxxxxxx00000101110111000000)

Bit	Reset	Description
19:0	0x5dc0	VALUE

28.10.49 HDMI_NV_PDISP_SOR_AUDIO_GEN_CTRL_0

This register only takes effect when it is written by software. The initialized value is not used.

DEV_ID: Device ID to identify the current chip.

REV_ID: Rev ID for the codec. This value is only used by the codec once this register has been written. Otherwise the default value will be used.

Offset: 0xc7 │ Byte Offset: 0x31c │ Read/Write: R/W │ Reset: 0x00220001 (0b00000000000100010xxxxxxx00000001)

Bit	Reset	Description
31:16	0x22	DEV_ID
7:0	0x1	REV_ID

28.10.50 HDMI_NV_HDACODEC_AUDIO_GEN_CTL_0

CHSTS_FS_3840

Currently the 4-bit coding for 384 kHz sampling rate is not available in the IEC-61937 specification. If the specification defines this value later, software needs to write the value during device initialization.

COPY_POLARITY: The polarity of the COPY bit is currently inverted in hardware as done in previous Tegra devices.

Offset: 0xd5 │ Byte Offset: 0x354 │ Read/Write: R/W │ Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx01111)

Bit	Reset	Description
4	OLD	COPY_POLARITY: 0 = OLD 1 = NEW
3:0	0xf	CHSTS_FS_3840

[THIS PAGE INTENTIONALLY LEFT BLANK]

29.0 HDMI CEC

The HDMI Consumer Electronics Control (CEC) block supports CEC standard communication over an HDMI connection. It support both remote control of HDMI devices attached to the Tegra® 4 device, and also allows other HDMI devices to control Tegra functions. Refer to the CEC appendix of the HDMI specification for details of this link.

29.1 Functional Description

The CEC module is an APB slave. It is located at address NV_ADDRESS_MAP_APB_CEC_BASE (0x70015000).

The CEC consists of hardware state machines that send and receive bytes over the CEC wire. Its simple host interface allows one byte to be sent and received at a time.

The CEC has interrupts for interrupt-driven input and output, and also permits polling I/O.

29.2 Programming Guidelines

29.2.1 Initialization

29.2.1.1 Clock and Reset

The CEC uses a fixed clock source – the APB bus clock. It also has a clock enable and reset in the CAR block.

- CAR.RST_DEVICES_W.SWR_CEC_RST = DISABLE
- CAR.CLK_OUT_ENB_W.CLK_ENB_CEC = ENABLE

29.2.1.2 Interrupt

The CEC is on PRI interrupt controller bit 3.

CEC INT_MASK should be enabled for TX_* and RX_* interrupts. TX_REGISTER_EMPTY and RX_REGISTER_FULL are the key interrupts; most others are errors.

Refer to the Interrupt Controller section for more information.

29.2.1.3 Pin Mux

The CEC is available on the HDMI_CEC pin. HDMI_CEC pin mux should be configured for:

- PM=0 (primary CEC function)
- PUPD=NORMAL
- TRISTATE=NORMAL
- E_INPUT=ENABLE
- OD=ENABLE

29.2.2 CEC Timing

Timing parameters are in units of 32 microseconds. See the HDMI specification for CEC timing requirements. In Tegra 4 devices, the registers have default (reset) values as indicated in the following table.

Register	Field	Value	Comment
RX_TIMING_0	RX_START_BIT_MAX_LO_TIME	0x7a	
	RX_START_BIT_MIN_LO_TIME	0x6d	
	RX_START_BIT_MAX_DURATION	0x93	
	RX_START_BIT_MIN_DURATION	0x86	
RX_TIMING_1	RX_DATA_BIT_MAX_LO_TIME	0x35	
	RX_DATA_BIT_SAMPLE_TIME	0x21	
	RX_DATA_BIT_MAX_DURATION	0x56	
	RX_DATA_BIT_MIN_DURATION	0x40	
RX_TIMING_2	RX_END_OF_BLOCK_TIME	0x50	
TX_TIMING_0	TX_START_BIT_LO_TIME	0x74	
	TX_START_BIT_DURATION	0x8d	
	TX_BUS_XITION_TIME	0x8	
	TX_BUS_ERROR_LO_TIME	0x71	
TX_TIMING_1	TX_LO_DATA_BIT_LO_TIME	0x2f	
	TX_HI_DATA_BIT_LO_TIME	0x13	
	TX_DATA_BIT_DURATION	0x4b	
	TX_ACK_NAK_BIT_SAMPLE_TIME	0x21	
TX_TIMING_2	BUS_IDLE_TIME_ADDITIONAL_FRAME	0x7	
	BUS_IDLE_TIME_NEW_FRAME	0x5	
	BUS_IDLE_TIME_RETRY_FRAME	0x3	

29.2.3 CEC Control

The CEC has several modes but only some are simulated or supported. TX_RX_MODE=ENABLE should be set last.

Register	Field	Value	Comment
SW_CONTROL	MODE	DISABLE	Use HW mode
INPUT_FILTER	FIFO_LENGTH	0	
	MODE	DISABLE	
HW_CONTROL	RX_LOGICAL_ADDRS	*Slave addresses (bitmap)*	If slave, holds addresses
	RX_SNOOP	DISABLE	
	RX_NAK_MODE	BLOCK	
	TX_NAK_MODE	BLOCK	
	FAST_SIM_MODE	DISABLE	SW should use DISABLE

Register	Field	Value	Comment
	TX_RX_MODE	ENABLE	Enable last
INT_MASK	TX_*, RX_*	DISABLE/ENABLE	ENABLE to enable interrupt

29.2.4 Transmission

Each message is a series of bytes, and each byte has several flags associated with it.

Register	Field	Value	Comment
TX_REGISTER	DATA	*byte*	8-bit byte
	EOM	1 on last byte	End of message flag
	ADDRESS_MODE	DIRECT/BROADCAST	
	GENERATE_START_BIT	1	
	RETRY_FRAME	0/1	

In general, the first byte contains source and destination addresses, per the CEC specification. The source address is chosen by software as the Tegra address. The destination address is either 15 for broadcast, or less than 15 for a particular slave. The last byte contains EOM=1. When sending to address 15 (broadcast), ADDRESS_MODE must be BROADCAST so that ACK and NAK have correct polarity. If this transmission is a retry of a previous one, RETRY_FRAME should be 1 so that correct retry timing is used.

The TX register interface is unusual in that the interrupt (TX_REGISTER_EMPTY) controls transmission – it's not merely a status signal. To transmit a byte, TX_REGISTER is first written with data, then the INT_STAT.TX_REGISTER_EMPTY interrupt must be cleared. If interrupt service routines are used, they must be careful when clearing an interrupt, because doing so incorrectly could cause a spurious or lost byte.

Transmission pseudo-code:

```
Transmit(unsigned char data, bool eom?, bool broadcast?):
        // enqueue byte
        Write TX_REGISTER: DATA = data,
                EOM = eom?,
                ADDRESS_MODE = broadcast?,
                GENERATE_START_BIT = 1
        // clear interrupt to enable transmission
        Write INT_STAT: TX_REGISTER_EMPTY = 1
        // wait for transmission
        Either Poll INT_STAT.TX_* to go high, or wait for TX_* interrupt.
        If TX_REGISTER_EMPTY=1, transfer is complete.
        If other TX_* interrupts set, then error.
```

Transmission of multi-block frames requires the blocks (bytes) to be sent back-to-back with no idle space between. If software does not provide the bytes in time, hardware will signal a TX_REGISTER_UNDERRUN interrupt and halt transmission (see Error Recovery below).

29.2.5 Reception

The RX_REGISTER_FULL interrupt indicates that a byte has been received and blocks further reception until cleared. If a subsequent byte is received while RX_REGISTER_FULL is high, an RX_REGISTER_OVERRUN error results.

When a byte is received, two additional EOM and ACK flags are provided as well.

Register	Field	Value	Comment
RX_REGISTER	DATA	*byte*	8-bit byte
	EOM	1 on last byte	End of message flag
	ACK	*bit from bus*	Expected value depends on whether broadcast or direct

Once INT_STAT indicates a byte has been received, first read the byte (and flags), and then clear the interrupt.

Reception pseudo-code:

```
Receive( unsigned char &data, bool &eom, bool &ack):
        // wait for data
        Either Poll INT_STAT.RX_REGISTER_FULL, or
               wait for CEC RX_REGISTER_FULL interrupt
        If INT_STAT.RX_* error interrupts set, then error
        data, eom, ack = Read RX_REGISTER.DATA/EOM/ACK
        // clear interrupt to allow further reception
        Write INT_STAT.RX_REGISTER_FULL = 1
```

29.2.6 Error Recovery

In case of transmission error such as UNDERRUN or NAKD, the transmission will stop. Alternatively, the soft reset sequence should be used:

```
tmp = CEC_HW_CONTROL
CEC_HW_CONTROL = 0
CEC_INT_STATUS = 0xffffffff
CEC_HW_CONTROL = tmp
```

29.3 CEC Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

29.3.1 CEC_SW_CONTROL_0

SW controlled mode is not supported. Leave disabled.

Offset: 0x0 | Read/Write: R/W | Reset: 0x000000XX (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	RW	DISABLE	MODE: 0 = DISABLE 1 = ENABLE
4	RO	X	FILTERED_RX_DATA_PIN
0	RO	X	RAW_INPUT_DATA_PIN

29.3.2 CEC_HW_CONTROL_0

The following steps are the best way to reset the CEC engine:

1. Set the CEC_HW_CONTROL_TX_RX_MODE to DISABLE
2. Set all the interrupt enable bits in CEC_HW_INTR_EN to DISABLE

3. Wait 1 second (1 second is the maximum CEC bus timeout period from the HDMI specification)
4. Set the CEC_HW_CONTROL_TX_RX_MODE to ENABLE and restart.

Other devices on the bus might have been communicating just fine. When the restart has completed, there might be a few spurious false start bits and other receive bus anomalies before normal operation is resumed.

The CEC_HW_CONTROL register contains the control bits and fields for operating and configuring the HW CEC engine.

Note:	Once the block is enabled, software should not attempt to change these configuration parameters without first disabling the block, except for the RX_LOGICL_ADDRS and RX_SNOOP fields.

RX_LOGICAL_ADDRS

The HDMI specification permits a single physical entity to claim logical addresses reserved for different functions (e.g., "Tuner x" and "Playback device y"). For a PC, it is entirely possible to be a multi-function device, so hardware might need to respond to 2 (or more) logical addresses (as many combinations as make sense). The next fields are used to configure the logical addresses that hardware will respond to. As per the specification, hardware will always respond to the broadcast address. This is a 15-bit field, where each bit position is associated with one of the logical addresses. A '1' in any bit position causes the hardware to respond to the associated logical address (i.e., capture data blocks/frames addressed to the address and properly ack/nak said blocks). To keep things simple, the mapping of bit position to logical address is direct, i.e., bit N maps to logical address N.

RX_SNOOP

When enabled, the hardware will intercept and forward to software all traffic on the CEC bus. It will continue to ack/nak only those addresses that are assigned to it.

RX_NAK_MODE

The HDMI specification is a bit unclear about what the initiator is supposed to do if an intermediate block of a frame is NAK’d, i.e., should the initiator cease transmitting immediately or complete the rest of the frame. The HDMI specification says that a follower may NAK a frame at any time, but the concern is that some initiator might not recover from such an early NAK’d frame. This bit will permit the receive state machine to operate in either mode. The choices are:

- BLOCK which means that a frame will be NAK’d as soon as a RX_REGISTER_OVERRUN is detected
- FRAME which means that the hardware will keep track of any RX_REGISTER_OVERRUNs that occur during frame reception, and NAK only during the last block.

TX_NAK_MODE

The HDMI specification is a bit unclear about what the initiator is supposed to do if an intermediate block of a frame is NAK’d, i.e., should the initiator cease transmitting immediately or complete the rest of the frame. This bit will permit the transmit state machine to operate in either mode. The choices are:

- BLOCK which means transmission will cease at the first NAK'd block
- FRAME which means that transmission will always continue to the end of the frame.

TX_RX_INTERRUPT_ROUTING

The engine generates a single composite interrupt output signal which can then be routed to either PMU or HOST. This bit controls that routing.

TX_RX_MODE

This bit enables or disables the operation of the HW CEC engine. If the TX_RX_MODE is changed to DISABLE, then the HW engine returns to the IDLE state irrespective of what it is doing and drives a 1 onto the CEC bus. If CEC_SW_CONTROL_MODE is ENABLED, the HW CEC engine operation is automatically disabled.

Offset: 0x4 | Read/Write: R/W | Reset: 0x0000XXXX (0b00xxxxx0xxxxxxx00xxxxxxxxxxxxxx)

Bit	Reset	Description
31	DISABLE	TX_RX_MODE: 0 = DISABLE 1 = ENABLE
30	DISABLE	FAST_SIM_MODE: 0 = DISABLE 1 = ENABLE
24	BLOCK	TX_NAK_MODE: 0 = BLOCK 1 = FRAME
16	BLOCK	RX_NAK_MODE: 0 = BLOCK 1 = FRAME
15	DISABLE	RX_SNOOP: 0 = DISABLE 1 = ENABLE
14:0	X	RX_LOGICAL_ADDRS: All logical addresses that should be matched. One bit per address.

29.3.3 CEC_INPUT_FILTER_0

The CEC_INPUT_FILTER register is used to configure the HW filtering that is required to deglitch the incoming receive data line. As data arrives into the chip, it is pushed into a 1 bit wide FIFO with a maximum of 64 entries deep. The actual used depth of the FIFO is set using the CEC_INPUT_FILTER_FIFO_LENGTH field. The used length is equal to CEC_INPUT_FILTER_FIFO_LENGTH+1. A datum is pushed into the FIFO once per microsecond tick. The FIFO bits should reset to '1' (the idle condition of the CEC bus).

The filtered output of the FIFO is computed roughly as follows:

```
filterMsk[63:0] = (1 << CEC_INPUT_FILTER_FIFO_LENGTH + 1) - 1;

if (reset || (CEC_INPUT_FILTER_MODE == CEC_INPUT_FILTER_MODE_DISABLE))
    fifo[63:0] = 0xffffffffffffffff;
else if (clock tick time)
    fifo[63:0] = (fifo[63:0] << 1) | rawInputDataPin;


if (reset || (CEC_INPUT_FILTER_MODE == CEC_INPUT_FILTER_MODE_DISABLE))
    filteredRxDataPin = 1;
else if ((clock tick time) && ((fifo & filterMsk) == filterMsk))
    filteredRxDataPin = 1;
else if ((clock tick time) && ((~fifo & filterMsk) == filterMsk))
    filteredRxDataPin = 0;
else
    filteredRxDataPin = filteredRxDataPin;
```

This logic should make sure that the filteredRxDataPin output only transitions to 0 or 1 when all the examined bits in the FIFO are also 0 or 1. Thus the filtered data line will not transition as long as there are glitches in the FIFO. The length of 64 provides a maximum operational length of 64 µs.

Every state transition on the filtered receive data line is reportable to software via the interrupt register described later.

Offset: 0x8 | Read/Write: R/W | Reset: 0x000000XX (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	DISABLE	MODE: 0 = DISABLE 1 = ENABLE
5:0	X	FIFO_LENGTH

29.3.4 CEC_SPARE_0

Spare register for future use.

Offset: 0xc | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SPARES

29.3.5 CEC_TX_REGISTER_0

CEC_HW_TX_REGISTER register

This register is used by software to provide the hardware with the actual data to be transmitted on the bus. Note that hardware will 'blindly' transmit what it is given. For example, it will not check to be sure that a proper legal initiator address has been provided, it will not check to be sure that the maximum frame length of 16 is not violated, etc. These higher level protocol checks are the domain of the SW. In order to facilitate easier software programming and smoother operation, hardware will make its own working copy of the fields in this register, thus quickly freeing up the register for software to write the next block. (Basically, it is a one-deep FIFO with fullness reported via the TX_REGISTER_EMPTY bit).

DATA

The actual 8 bits of address/data to be transmitted on the bus. The data is always transmitted MSB (bit 7) first.

EOM

This is the "end of message" bit for this block of data. This bit is set at the last block of the frame.

ADDRESS_MODE

This bit indicates to the hardware whether this particular block is directly addressed or broadcast addressed (which in turn dictates how hardware is to interpret the ACK/NAK for the block).

GENERATE_START_BIT

Indicates to the hardware that it should precede the transmission of this block of data with a start bit.

RETRY_FRAME

Indicates if the current frame is a retry frame or not. Based on this value, hardware chooses an appropriate bus idle time programmed in TX_TIMING2 register and waits for the bus to be idle before it can attempt to send a start bit. This bit is only meaningful when TX_GENERATE_START_BIT is set.

RETRY FRAME	LAST_FRAM_ SENT_BY_US	WAIT TIME
1	YES	TIMING2_BUS_IDLE_TIME_RETRY_FRAME
1	NO	TIMING2_BUS_IDLE_TIME_RETRY_FRAME
0	NO	TIMING2_BUS_IDLE_TIME_NEW_FRAME
0	YES	TIMING2_BUS_IDLE_TIME_ADDITIONAL_FRAME

If a frame is being sent immediately following the previous frame, hardware waits for "TIMING2_BUS_IDLE_TIME_ADDITIONAL_FRAME". However, if someone else uses the bus before that time elapses, the hardware resets its wait time counter and waits for "TIMING2_BUS_IDLE_TIME_NEW_FRAME".

Offset: 0x10 | Read/Write: R/W | Reset: 0x00010000 (0bxxxxxxxxxxxxxxx01xxx0xxx000000000)

Bit	Reset	Description
17	DISABLE	RETRY_FRAME: 0 = DISABLE 1 = ENABLE
16	ENABLE	GENERATE_START_BIT: 0 = DISABLE 1 = ENABLE
12	DIRECT	ADDRESS_MODE: 0 = DIRECT 1 = BROADCAST
8	0x0	EOM
7:0	0x0	DATA

29.3.6 CEC_RX_REGISTER_0

CEC_HW_RX_REGISTER register

This register is used by hardware to buffer data to the software. When a block of data has been received from the bus, it is stored here for software (and the RX_REGISTER_FULL bit is set). The hardware also has a 'working copy' of the receive register to give software as much time as possible to empty it. When a full block is assembled, hardware places it into RX_REGISTER, so the register is like a 1 deep FIFO.

DATA

The 8 bits of data that were read from the bus.

EOM

The EOM bit read from the bus.

ACK_NAK

The ACK/NAK bit as read from the bus. In some cases (broadcast block) even though the device may have ACK'd the frame, some other device on the bus may NAK the frame, and software would need to know this.

Offset: 0x14 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
9	X	ACK
8	X	EOM
7:0	X	DATA

29.3.7 CEC_RX_TIMING_0_0

CEC_HW_RX_TIMING0 register

Note: All timing registers must be configured before the HW CEC engine is enabled.

RX_START_BIT_MAX_LO_TIME

nominal 3.9 ms, (3900/32 = 122 intervals) = 3.904 ms

The maximum time the received bus can remain low, and still be considered the beginning of a proper start bit. If the start bit low time overruns this time, the start bit is considered invalid and is ignored.

RX_START_BIT_MIN_LO_TIME

nominal 3.5 ms, (3500/32 = 109 intervals) = 3.488 ms

The minimum time the received bus can remain low and still be considered the beginning of a proper start bit. If the start bit low time underruns this time, the start bit is considered invalid and is ignored.

RX_START_BIT_MAX_DURATION

nominal 4.7 ms, (4700/32 = 147 intervals) = 4.704 ms

The maximum total duration of the start bit from the first detected high-to-low transition to the high-to-low transition that begins the first address bit. If the start bit duration overruns this time, the start bit is considered invalid and is ignored.

RX_START_BIT_MIN_DURATION

nominal 4.3 ms (4300/32 = 134 intervals) = 4.288 ms

The minimum total duration of the start bit from the first detected high-to-low transition to the high-to-low transition that begins the first address bit. If the start bit duration underruns this time, the start bit is considered invalid and is ignored.

Offset: 0x18 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	RX_START_BIT_MIN_DURATION
23:16	X	RX_START_BIT_MAX_DURATION
15:8	X	RX_START_BIT_MIN_LO_TIME
7:0	X	RX_START_BIT_MAX_LO_TIME

29.3.8 CEC_RX_TIMING_1_0

CEC_HW_RX_TIMING1 register

Note: All timing registers must be configured before the HW CEC engine is enabled.

RX_DATA_BIT_MAX_LO_TIME

nominal 1.7 ms, (1700/32 = 53 intervals) = 1.696 ms

The maximum time the received bus can remain low and still be considered the beginning of a proper start bit. If the start bit low time overruns this time, the start bit is considered invalid and is ignored.

RX_DATA_BIT_SAMPLE_TIME

nominal 1.05 ms, (1050/32 = 33 intervals) = 1.056 ms

The time to sample that data bit.

RX_DATA_BIT_MAX_DURATION

nominal 2.75 ms, (2750/32 = 86 intervals) = 2.752 ms

The maximum total duration of the start bit from the first detected high-to-low transition to the high-to-low transition that begins the first address bit. If the start bit duration overruns this time, the start bit is considered invalid and is ignored.

RX_DATA_BIT_MIN_DURATION

nominal 2.05 ms (2050/32 = 64 intervals) = 2.048 ms

The minimum total duration of the start bit from the first detected high-to-low transition to the high-to-low transition that begins the first address bit. If the start bit duration underruns this time, the start bit is considered invalid and is ignored.

Offset: 0x1c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	RX_DATA_BIT_MIN_DURATION
23:16	X	RX_DATA_BIT_MAX_DURATION
15:8	X	RX_DATA_BIT_SAMPLE_TIME
7:0	X	RX_DATA_BIT_MAX_LO_TIME

29.3.9 CEC_RX_TIMING_2_0

CEC_HW_RX_TIMING2 register

Note: All timing registers must be configured before the HW CEC engine is enabled.

RX_END_OF_BLOCK_TIME

nominal 1.9 ms, (1900/32 = 80 intervals) = 1.9 ms

The time to wait after the start of the final ACK/NAK phase of frame transmission before returning to the idle state. (Needed since the last ACK/NAK bit will not be followed by another high-to-low transition.)

Offset: 0x20 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	RX_END_OF_BLOCK_TIME

29.3.10 CEC_TX_TIMING_0_0

The next set of timing registers configures the CEC logic for HW assisted operation. To permit flexible configuration (to support possible semi-compliant devices), the bit timing control and check values are programmable (rather than just directly using the specification values).

While the block itself works on a 1 µs tick, in order to save register space, these timing numbers are specified in units of 32 µs.

First, the registers for the various timing intervals (there are quite a few of them).

CEC_HW_TX_TIMING0 register

Note: All timing registers must be configured before the HW CEC engine is enabled.

For details of exact values and tolerances, refer to the CEC appendix of the HDMI specification.

TX_START_BIT_LO_TIME

nominal 3.7 ms, (3700/32 = 116 intervals) = 3.712 ms

How long to hold the bus low during the start bit.

TX_START_BIT_DURATION

nominal 4.5 ms, (4500/32 = 141 intervals) = 4.512 ms

The total duration of the start bit

TX_BUS_XITION_TIME

nominal 250 µs, (250/32 = 8 intervals) = 256 µs

Time to wait for bus to settle after transitioning output.

TX_BUS_ERROR_LO_TIME

nominal 3.6 ms (3600/32 = 113 intervals) = 3.616 ms

Time to drive bus low when bus error must be signaled on the bus

Offset: 0x24 | Read/Write: R/W | Reset: 0xXX0XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	TX_BUS_ERROR_LO_TIME
19:16	X	TX_BUS_XITION_TIME
15:8	X	TX_START_BIT_DURATION
7:0	X	TX_START_BIT_LO_TIME

29.3.11 CEC_TX_TIMING_1_0

CEC_HW_TX_TIMING1 register

Note: All timing registers must be configured before the HW CEC engine is enabled.

TX_LO_DATA_BIT_LO_TIME

nominal 1.5 ms, (1500/32 = 47 intervals) = 1.504 ms

How long to hold the bus low when transmitting a '0'.

TX_HI_DATA_BIT_LO_TIME

nominal 0.6 ms, (600/32 = 19 intervals) = 6.08 ms

How long to hold the bus low when transmitting a '1'.

TX_DATA_BIT_DURATION

nominal 2.4 ms, (2400/32 = 75 intervals) = 2.4 ms

The total duration of a data or ACK/NAK bit.

TX_ACK_NAK_BIT_SAMPLE_TIME

nominal 1.05 ms, (1050/32 = 33 intervals) = 1.056 ms

The time to sample that ACK/NAK bit.

Offset: 0x28 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	TX_ACK_NAK_BIT_SAMPLE_TIME

Bit	Reset	Description
23:16	X	TX_DATA_BIT_DURATION
15:8	X	TX_HI_DATA_BIT_LO_TIME
7:0	X	TX_LO_DATA_BIT_LO_TIME

29.3.12 CEC_TX_TIMING_2_0

CEC_HW_TX_TIMING2 register

This register contains the bus idle time values for the various scenarios indicated in the HDMI specification (section 9.1). Software indicates to the hardware the amount of delay in units of data bit periods, as defined by the CEC_HW_TX_TIMING1_TX_DATA_BIT_DURATION field.

ADDITIONAL_FRAME:

The amount of time the bus needs to be idle before a new frame can be sent immediately after sending a frame. The suggested value is >= 7.

NEW_FRAME:

The amount of time the bus needs to be idle before a new frame can be sent if we were not the initiator for the previous frame. The suggested value is >= 5.

RETRY_FRAME:

The amount of time the bus needs to be idle before the same frame can be resent. The suggested value is >= 3.

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
11:8	X	BUS_IDLE_TIME_RETRY_FRAME
7:4	X	BUS_IDLE_TIME_NEW_FRAME
3:0	X	BUS_IDLE_TIME_ADDITIONAL_FRAME

29.3.13 CEC_INT_STAT_0

CEC_HW_INTR register

This register contains the individual interrupt and status bits for the CEC HW unit. The CEC_HW_INTR_EN register contains the corresponding enable bit for each interrupt/status bit in the interrupt register. The interrupt enable determines whether the interrupt propagates to the PMIC as an interrupt, but the interrupt status bit itself is always set if the described condition occurs. Software writes a '1' to any interrupt/status bit in order to clear it.

Because the HW state machine also looks at the state of the interrupt bits when determining what action(s) to take, it is important for software to operate in roughly the following order when processing interrupts from this source:

1. Read the interrupt register to determine what caused the interrupt and what needs to be done.
2. Perform the operations required, e.g., reload TX register, read RX register.
3. Finally, clear the corresponding interrupt bits.

TX_REGISTER_EMPTY

The transmit register is empty, so software is free to write the next block of the frame into the register. For multi-block frames, software must in fact provide the next block before the current block is sent, otherwise a TX_REGISTER_UNDERRUN error will occur. When the TX engines reads from the TX_REGISTER register and stores a local copy, this interrupt is asserted.

TX_REGISTER_UNDERRUN

The transmit register was empty at the time the transmit hardware needed to have the next block ready to send. This is an error condition. The transmitter will have stopped transmitting, and recovery will require resending the entire frame from scratch.

TX_FRAME_OR_BLOCK_NAKD

Hardware sets this bit if any block/frame is NAK'd. If software sees this bit set, it knows it will have to resend the frame later.

TX_ARBITRATION_FAILED

Hardware sets this bit during the address block phase of transmitting a frame, if failed to win arbitration. In this case, transmission will cease, and the HW will flush any pending data in the TX_REGISTER. Software will need to retry the frame from scratch (after the appropriate signal free time, i.e., hardware will not automatically retry).

TX_BUS_ANOMALY_DETECTED

Sometime during block transmission, hardware detected anomalous behavior on the bus (e.g., the bus remained low after transmitter had signaled high). When such an anomaly is detected, the transmitter ceases operation, releases the bus high, and flushes any pending data in the TX_TRANSMIT register.

TX_FRAME_TRANSMITTED

This will be asserted at the end of last data bit, i.e., the ACK bit of the last block is sent. This is just an indication that the last block is sent but does not necessarily mean the transmission was successful. Software should still look for other interrupts to check for any errors.

RX_REGISTER_FULL

The hardware has assembled a complete block from the bus and placed it into the RX_REGISTER. Software should read the block immediately to ensure the RX_REGISTER is empty before hardware has the next block ready to load. When the HW RX engine writes RX data to the RX_REGISTER this bit is set.

RX_REGISTER_OVERRUN

Software failed to read the RX_REGISTER before hardware had another block of data ready to transfer. Hardware will NAK the block/frame, and the initiator will need to retry later. Software will need to flush the partially accumulated frame.

RX_START_BIT_DETECTED

Whenever the receive engine detects a start bit, it will set this bit. This bit is used for debug.

RX_BUS_ANOMALY_DETECTED

The receiver detected some anomaly (including bus errors) on the bus. A bus error has a specific definition in the HDMI specification, but there are other kinds of anomalies that can occur. All anomalies and bus errors are reported here.

RX_BUS_ERROR_DETECTED

The receiver has detected the specific anomaly called a bus error and then signaled a bus error on the bus per the HDMI specification.

FILTERED_RX_DATA_PIN_TRANSITION_H2L

In direct SW bus drive mode, software needs to be interrupted whenever the state of the received data line transitions. This interrupt is asserted on a 1 to 0 transition.

FILTERED_RX_DATA_PIN_TRANSITION_L2H

In direct SW bus drive mode, software needs to be interrupted whenever the state of the received data line transitions. This interrupt is asserted on a 0 to 1 transition

Offset: 0x30 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14	X	FILTERED_RX_DATA_PIN_TRANSITION_L2H: 0 = INACTIVE 1 = ACTIVE
13	X	FILTERED_RX_DATA_PIN_TRANSITION_H2L: 0 = INACTIVE 1 = ACTIVE
12	X	RX_BUS_ERROR_DETECTED: 0 = INACTIVE 1 = ACTIVE
11	X	RX_BUS_ANOMALY_DETECTED: 0 = INACTIVE 1 = ACTIVE
10	X	RX_START_BIT_DETECTED: 0 = INACTIVE 1 = ACTIVE
9	X	RX_REGISTER_OVERRUN: 0 = INACTIVE 1 = ACTIVE
8	X	RX_REGISTER_FULL: 0 = INACTIVE 1 = ACTIVE
5	X	TX_FRAME_TRANSMITTED: 0 = INACTIVE 1 = ACTIVE
4	X	TX_BUS_ANOMALY_DETECTED: 0 = INACTIVE 1 = ACTIVE
3	X	TX_ARBITRATION_FAILED: 0 = INACTIVE 1 = ACTIVE
2	X	TX_FRAME_OR_BLOCK_NAKD: 0 = INACTIVE 1 = ACTIVE
1	X	TX_REGISTER_UNDERRUN: 0 = INACTIVE 1 = ACTIVE
0	X	TX_REGISTER_EMPTY: 0 = INACTIVE 1 = ACTIVE

29.3.14 CEC_INT_MASK_0

Mask register for interrupts. When a field in this register is set to ENABLE, the corresponding field with a value of 1 in the INT_STATUS register will result in assertion of an interrupt line.

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx0000000xx000000)

Bit	Reset	Description
14	DISABLE	FILTERED_RX_DATA_PIN_TRANSITION_L2H: 0 = DISABLE 1 = ENABLE
13	DISABLE	FILTERED_RX_DATA_PIN_TRANSITION_H2L: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	DISABLE	RX_BUS_ERROR_DETECTED: 0 = DISABLE 1 = ENABLE
11	DISABLE	RX_BUS_ANOMALY_DETECTED: 0 = DISABLE 1 = ENABLE
10	DISABLE	RX_START_BIT_DETECTED: 0 = DISABLE 1 = ENABLE
9	DISABLE	RX_REGISTER_OVERRUN: 0 = DISABLE 1 = ENABLE
8	DISABLE	RX_REGISTER_FULL: 0 = DISABLE 1 = ENABLE
5	DISABLE	TX_FRAME_TRANSMITTED: 0 = DISABLE 1 = ENABLE
4	DISABLE	TX_BUS_ANOMALY_DETECTED: 0 = DISABLE 1 = ENABLE
3	DISABLE	TX_ARBITRATION_FAILED: 0 = DISABLE 1 = ENABLE
2	DISABLE	TX_FRAME_OR_BLOCK_NAKD: 0 = DISABLE 1 = ENABLE
1	DISABLE	TX_REGISTER_UNDERRUN: 0 = DISABLE 1 = ENABLE
0	DISABLE	TX_REGISTER_EMPTY: 0 = DISABLE 1 = ENABLE

29.3.15 CEC_HW_DEBUG_RX_0

Offset: 0x38 | Read/Write: RO | Reset: 0x0XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
27	X	RXDATABIT_SAMPLE_TIMER
26	X	LOGICADDR_MATCH
25	X	FORCELOOUT
24:21	X	STATE
20:17	X	RXBIT_COUNT
16:0	X	DURATION_COUNT

29.3.16 CEC_HW_DEBUG_TX_0

Offset: 0x3c | Read/Write: RO | Reset: 0x0XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26	X	TXDATABIT_SAMPLE_TIMER
25	X	FORCELOOUT
24:21	X	STATE

Bit	Reset	Description
20:17	X	TXBIT_COUNT
16:0	X	DURATION_COUNT

29.3.17 CEC_HW_SPARE_0_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SCRATCH

30.0 MIPI-CSI (CAMERA SERIAL INTERFACE)

This section describes the Camera Serial Interface (CSI), which is based on the MIPI CSI 2.0 standard specification. The MIPI CSI 2.0 standard is available from MIPI to its members at http://www.mipi.org/.

30.1 Functional Description

The CSI implements a MIPI compliant CSI receiver that may receive data from an external camera module with a CSI transmitter. The CSI is designed to provide for direct connection to 3 camera modules and allow any two of the modules to be active simultaneously:

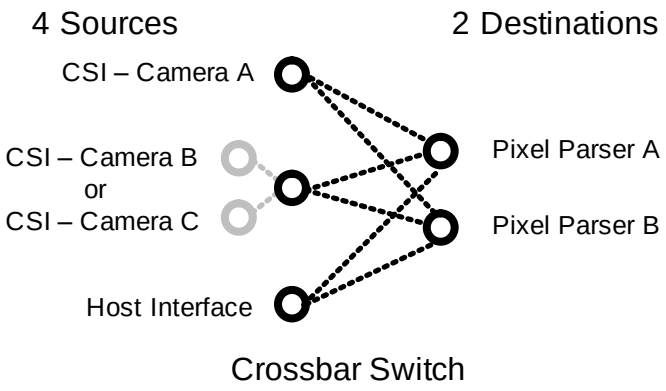
- CSI-A interface supports 1 clock lane and up to 4 data lanes for Camera A.
- CSI-B interface supports 1 clock lane and up to 4 data lanes for Camera B OR a single data lane for Camera C.

The CSI-B interface is shared between Camera A and Camera B, allowing only one or the other to be active at a time. The CSI offers both single and multi-camera configuration options. These options include a single camera configuration with a 1 to 4 lane sensor connected to CSI-A or CSI-B, a dual camera configuration with a 1 to 4 lane sensor connected to CSI-A and a 1 lane sensor to CSI-B, or a stereo camera configuration with a 1 to 4 lane sensor connected to CSI-A and another 1 to 4 lanes sensor to CSI-B.

In addition to direct connection to camera modules, the CSI interface can also receive a digital CSI compatible byte stream or payload data only from the host interface. The CSI stream coming from the host is provided to take advantage of the rich CSI data formats.

A maximum of two data/pixel streams can be processed simultaneously at any given time. The two streams can come from any one or two of the four possible sources, as shown in Figure 121. If the two streams come from a single source, then the streams are separated using a filter indexed on different virtual channel numbers or data types. In case of separation using data types, the normal data type is separated from the embedded data type. Because there are only two pixel parsers (PPA and PPB), the virtual channel and embedded data capability cannot be used at the same time.

Figure 121: Data Source/Destination Options



The CSI may receive data from camera modules through the MIPI serial or the host interfaces. A test pattern generator has been implemented to assist validation.

The CSI supports both single-shot mode and continuous mode.

The CSI is designed with error resilience.

30.2 Use Cases

The use cases supported by the CSI include basic single camera, multiple cameras, embedded data, virtual channel, and host. The still and video capture sequences involve different single camera and multi-camera configuration options. Table 98 summarizes the use case categories.

Table 98: Summary of CSI Camera Use Cases

Case	Description	Sources	Destinations
Single Camera Video Capture	Standard CSI Basic preview or video capture using single camera	Camera A,B, or C	PPA or PPB
Stereo Camera Video Capture	Stereo pair video capture	Camera A and B	PPA and PPB
Single Camera Still Capture	Basic high resolution still capture supporting single shot and burst capture	Camera A, B, or C	PPA or PPB
Stereo Camera Still Capture	Stereo camera high resolution still capture	Camera A and B	PPA and PPB
Dual Camera Video Capture*	Video feed of front facing camera and user facing camera for video annotation, video preview during Video Conference	Camera A or B, and C	PPA and PPB
Video Conference Mono Still Capture & Preview*	Video feed of user facing camera for video conference and mono still capture from front facing camera.	Camera A or B, and C	PPA and PPB
Virtual Channel	One stream goes to two parsers. Packet filtering based on VC number in header	Camera A,B,C, or Host	PPA and PPB
Embedded Data	One stream goes to one parser. Packet filtering based on data type in header	Camera A,B,C, or Host	PPA or PPB
Embedded Data	One stream goes to one parser. Has normal and embedded data	Camera A,B,C, or Host	PPA or PPB
Host	Payload-only or Packet from Host interface	Host I/F	PPA or PPB

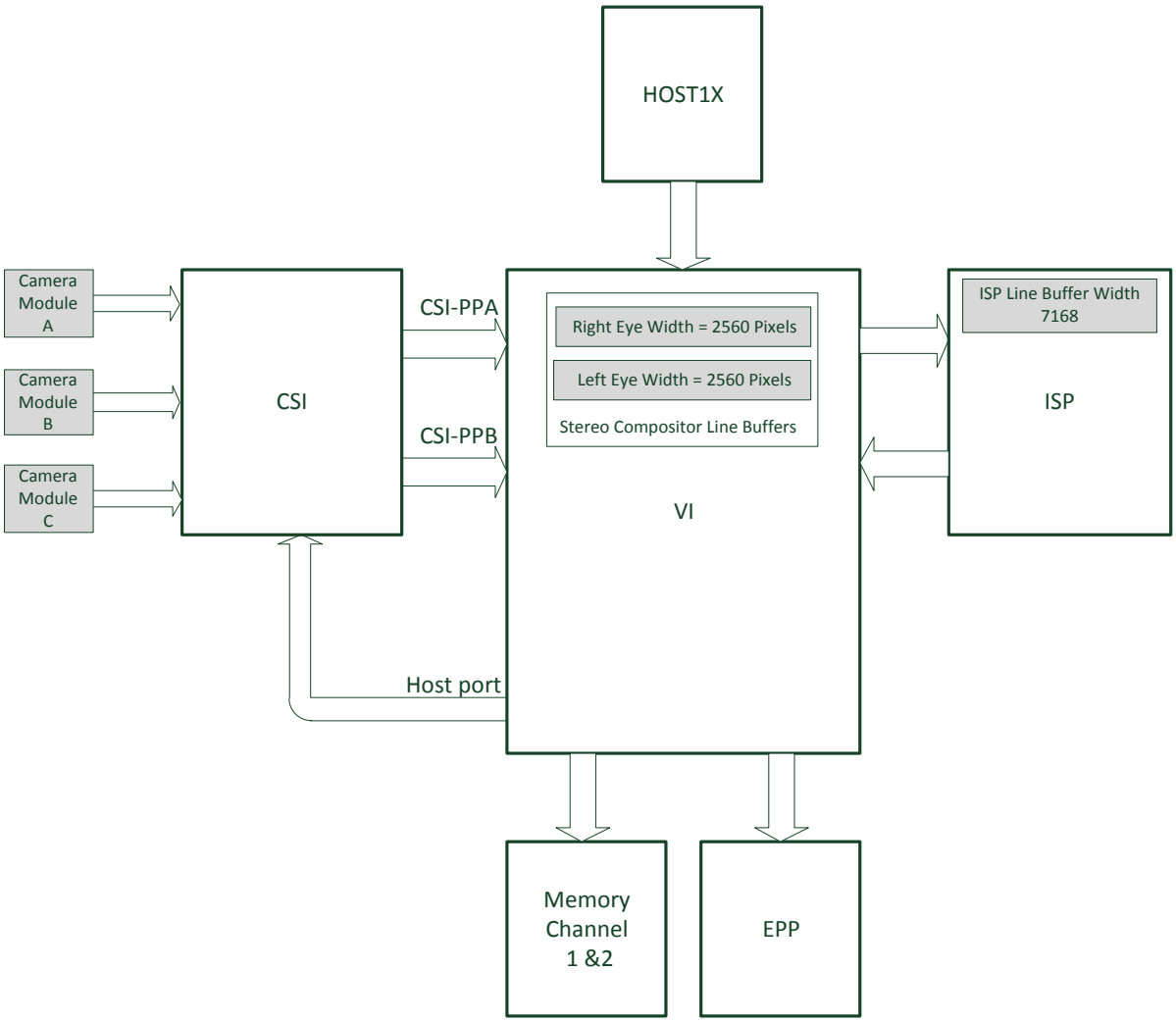
* Non-stereo dual camera models assume mixed data formats: Camera A – RAW10, Camera B - YCbCr 4:2:2.

The dual camera use cases require two capture streams from two different sensors, each imaging a separate scene. Within the Tegra® 4 camera system, it is necessary to process two different images with different algorithm sequences/ISP setup or to receive a preprocessed YCbCr image and deliver the stream to system memory and the other to the ISP.

30.2.1 Primary Mono and Stereo Camera Use Cases

The primary mono and stereo camera user models involve the capture of either pre-processed image data from a camera module in YCbCr 4:2:2 format or as a Raw Bayer stream with 10-bit precision. Additional pixel formats for image data may be accepted, but Raw 10-bit and YCbCr 4:2:2 are used primarily. Figure 122 illustrates the datapaths associated with the Tegra 4 camera subsystem.

Figure 122: Illustration of Datapath Through the Tegra 4 Camera Subsystem



Depending on the mode of operation, image resolution, and resulting pixel rate, one of several paths through the VI may be used. Table 25 summarizes the primary video and still captures use cases and the path through the VI.

Table 99: Data Through VI for Primary Camera Use Case

Use Cases	Data Received from Camera	Destination through VI
Mono Video Capture and Preview	Raw Bayer (10 bpp)	CSI -> VI -> ISP
	YCbCr 4:2:2 (16 bpp)	CSI -> VI -> MEMC or EPP
Stereo Video Capture and Preview	Raw Bayer (10 bpp)	CSI -> VI Stereo Comp -> ISP
	YCbCr 4:2:2 (16 bpp)	CSI -> VI -> MEMC or EPP
Mono Raw Bayer Still Capture	Raw Bayer (10 bpp)	CSI -> VI -> ISP
Stereo Raw Bayer Still Capture	Raw Bayer (10 bpp) Image A	CSI -> VI -> ISP
	Raw Bayer (10 bpp) Image B	CSI -> VI -> MEMC Host1X -> VI -> ISP

The raw Bayer mono and stereo video capture use cases are either processed by the ISP directly or are first buffered using the 2560 pixel stereo compositor prior to being presented to the ISP. Since the YCbCr 4:2:2 data has already been processed, it is likely that it will either be routed to memory channel 1 or 2 or EPP for both mono and stereo use cases.

The ISP can support from 300 to 350 MP/s peak throughput, depending on operating frequency, for raw 10-bit Bayer data and provides line buffering to manage up to 7168 pixel image width. During mono still image capture of raw 10-bit data, the data may be presented directly to the ISP through the VI as long as the frame rate and data rate are managed within the throughput constraint.

The stereo still capture model will require two high-resolution images to be captured simultaneously, resulting in a 2X increase in pixel data rate both into and out of the CSI. One of the stereo pair images may be processed through the ISP with the same limitations as with the mono still capture with the second stereo image being presented to memory through memory channel 1 or 2.

30.2.1.1 Video Capture and Preview Use Case

Table 100 defines the various camera configurations that the CSI can support for the primary HD video capture user models. The table summarizes the data rates for HD video capture when either raw 10-bit Bayer or processed YCbCr 4:2:2 data is being received by the CSI.

Table 100: Summary of Bayer (10-bit) and YCbCr 4:2:2 Video Capture Data Rates @ 60 fps

Input Format	Camera Video Capture Modes	Pixel Data Rate MPixels/s	MIPI Lanes Required @ Bit Rate		
			600 Mbps	800 Mbps	1 Gbps
Bayer 10 bpp	1280x720p	55.296	1	1	1
	1280x720p Stereo	110.592	2 (1+1)	2 (1+1)	2 (1+1)
	1920x1080p	124.416	3	2	2
	1920x1080p Stereo	248.832	6 (3+3)	4 (2+2)	4 (2+2)
YCbCr 4:2:2 16 bpp	1280x720p	55.296	2	2	1
	1280x720p Stereo	110.592	4 (2+2)	4 (2+2)	2 (1+1)
	1920x1080p	124.416	4	3	3
	1920x1080p Stereo	248.832	8 (4+4)	6 (3+3)	6 (3+3)

All raw 10-bit Bayer video capture sequences are presented to the ISP by the VI directly because the pixel data rate required is below 300 MP/s and the number of pixels per row is less than 7168 pixels. The stereo capture sequences will buffer a row of pixels from the left and right cameras within the VI prior to presenting the stereo composite super row to the ISP for processing. The stereo compositor within the VI provides up to 2560 pixels to be buffered for the right and left cameras.

30.2.1.2 Still Image Capture Use Case

The still image capture use case assumes that the ISP is first used to process the preview stream. When a still capture is indicated, the sensor is reconfigured, and a still frame is captured and presented to the ISP. Because the ISP is limited in throughput, higher resolution images may be captured and processed directly by the ISP at lower frame rates. For example, a 24 MP image may be captured at 12 fps instead of 15 fps with a 300 MP/sec throughput limitation at the expense of rolling shutter artifacts in case of scene or user motion. Similarly, with the increased pixel throughput at 350 MP/sec, the 24 MP image may be captured at 14 fps. The number of MIPI lanes required to support the peak data rate for various configurations will vary as a function of both the bit rate and pixel throughput. Table 101 summarizes the mono still capture pixel rate and the corresponding MIPI lanes for 600 Mbps, 800 Mbps, and 1 Gbps for maximum ISP throughput of 300 or 350 MP/s.

Table 101: Summary of 10-bit Bayer Mono Still Image Capture Data Rates

Sensor Size (MP)	HRES x VRES	Peak Data Rate (FPS)/(MP/s)		MIPI Lanes Required @ Peak FPS (FPS)/(Number of Lanes)					
				Max of 300 MP/s			Max of 350 MP/s		
		Max of 300 MP/s	Max of 350 MP/s	0.6	0.8	1.0	0.6	0.8	1.0
5.17	2592x1944	15/75.58	15/75.58	15/2	15/1	15/1	15/2	15/1	15/1
8	3264x2448	15/119.85	15/119.85	15/3	15/2	15/2	15/3	15/2	15/2
9	3488x2616	15/136.87	15/136.87	15/3	15/2	15/2	15/3	15/2	15/2
10	4320x2430	15/157.46	15/157.46	15/3	15/3	15/2	15/3	15/3	15/2
12.6	4224x3000	15/190.08	15/190.08	15/4	15/3	15/2	15/4	15/3	15/2
14.6	4416x3312	15/219.39	15/219.39	15/4	15/3	15/3	15/4	15/3	15/3
16	4928x3280	15/242.46	15/242.46	-	15/4	15/3	-	15/4	15/3
21	5616x3744	14/294.37	15/315.39	-	14/4	14/4	-	-	15/4
24	6048x4032	12/292.63	14/341.40	-	12/4	12/4	-	-	14/4
32	6464x4864	9/282.97	11/345.85	-	9/4	9/4	-	-	11/4

In Table 101, the dashed cells are lane configurations that cannot support the maximum pixel rate and are limited by a 15 fps capture target and peak data rate driven by the ISP throughput. These lane configurations are still possible; however, it is necessary to degrade the frame rate (and data rate) from the sensor so that the maximum bit rate supported by the MIPI configuration is not exceeded.

The stereo still image capture use case assumes the ISP is used to first process the image data for user scene composition. When indicated, the sensor is reconfigured to capture a high-resolution still image pair from the left and right camera modules. Still images 5.17 MP and greater exceed the 2560 line width for the stereo compositor and need to utilize system memory to buffer the raw Bayer data from one of the cameras. Table 102 summarizes the aggregate pixel data rate for the stereo still use case for maximum ISP throughput of 300 and 350 MP/s.

Table 102: Summary of 10-bit Bayer Stereo Still Image Capture Data Rates

Sensor Size (MP)	HRES x VRES	Stereo Peak Data Rate (FPS)/(MP/s)		MIPI Lanes Required @ Peak FPS (FPS)/(Number of Lanes Right + Number of Lanes Left)					
				Max of 300 MP/s			Max of 350 MP/s		
		Max of 300 MP/s	Max of 350 MP/s	0.6	0.8	1.0	0.6	0.8	1.0
5.17	2592x1944	15/151.16	15/151.16	15/2+2	15/1+1	15/1+1	15/2+2	15/1+1	15/1+1
8	3264x2448	15/239.70	15/239.70	15/3+3	15/2+2	15/2+2	15/3+3	15/2+2	15/2+2
9	3488x2616	15/273.74	15/273.74	15/3+3	15/2+2	15/2+2	15/3+3	15/2+2	15/2+2
10	4320x2430	15/314.92	15/314.92	15/3+3	15/3+3	15/2+2	15/3+3	15/3+3	15/2+2
12.6	4224x3000	15/380.16	15/380.16	15/4+4	15/3+3	15/2+2	15/4+4	15/3+3	15/2+2
14.6	4416x3312	15/438.78	15/438.78	15/4+4	15/3+3	15/3+3	15/4+4	15/3+3	15/3+3
16	4928x3280	15/484.92	15/484.92	-	15/4+4	15/3+3	-	15/4+4	15/3+3
21	5616x3744	14/588.74	15/630.79	-	14/4+4	14/4+4	-	-	15/4+4
24	6048x4032	12/585.26	14/682.80	-	12/4+4	12/4+4	-	-	14/4+4
32	6464x4864	9/565.94	11/691.70	-	9/4+4	9/4+4	-	-	11/4+4

The capture of the stereo high-resolution stills may be either first routed to system memory through the VI memory interface and the pair is processed in an offline mode while the preview is disabled or one image in the pair may be processed through the ISP while the other is passed to system memory. The image passed to system memory may be processed through the ISP in an offline mode prior to enabling the preview.

30.3 Input Data Format

The supported data formats are summarized in Table 103.

Table 103: Supported CSI Formats

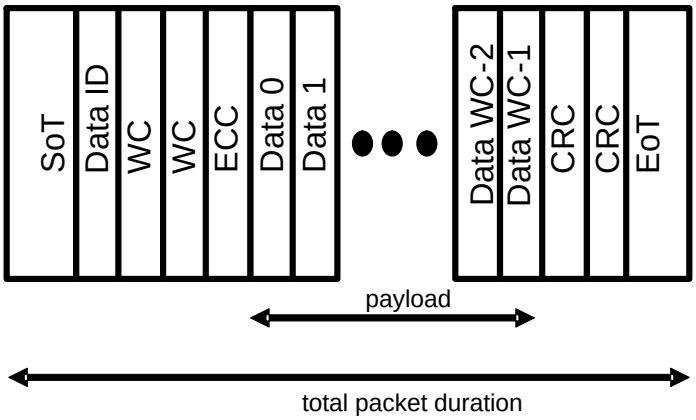
Data Format	
RGB	RGB888, RGB666, RGB565, RGB555, RGB444
YUV	YUV422-8b, YUV422-10b, YUV420-8b (legacy), YUV420-8b, YUV420-10b, YUV444-8b
RAW	RAW6, RAW7, RAW8, RAW10, RAW12, RAW14
User defined	JPEG8
Embedded	Embedded control information

The formats YUV422/420-10b are converted to YUV422/420-8b by taking the most significant bits.

30.4 CSI Packet Structure

The CSI packet structure is illustrated in Figure 123. It has a start-of-transmission sequence (SoT), which contains a 1-byte preamble sequence. The packet is composed of a 4-byte header, a body of word count (WC) bytes, and a 2-byte CRC check. The end-of-transmission (EoT) is indicated by the line state going from high-speed transmission to the LP11 state.

Figure 123: CSI Packet Structure



30.5 CSI Implementation

The CSI interface consists of:

- MIPI D-PHY block
- Control Interface Logic (CIL) for Serial Clock Receiver
- CSI datapath module which is implemented as a sub-module of VI module

In the Tegra 4 implementation, two CSI bricks including csia_pad with 2x data lanes and csib_pad with 2x data lane receive serial data from cameras, deserialize them into 8-bit parallel data, and send them to the CSICIL block. The CSICIL block performs packet boundary detection by searching for the packet preamble. The main CSI module receives an 8-bit packet aligned data from CSICIL, performs parity checking on the header, header decoding, CRC check, and pixel parsing. The output is sent out to the VI through the 28-bit bus, which outputs 1 pixel per VI clock.

30.6 Performance Limitations

The performance of the CSI interface is subject to several factors that must all be met. The performance factors that are directly related to the CSI interface are the serial data rate and the internal pixel rate. Maximum serial data rate is expected to be 1Gbps given the ideal system condition. This data rate may be degraded depending on system design and may also be limited by the CSI transmitter in the camera.

For each CSI data lane, the serial data stream received by the CSI PHY is internally converted to a byte stream by the CSI Control Interface Logic (CIL) before further processed by the CSI module that resides as part of the Video Input (VI) module. The CSI byte clock is used to transfer this data byte stream (1 byte per data lane) between the CIL and the CSI datapaths. This CSI byte clock is derived from the received CSI serial clock by dividing the CSI serial clock by 4. So for 1 Gbps serial CSI data rate, the serial clock frequency is 500 MHz and the byte clock frequency is 125 MHz.

The byte stream received by the CSI CIL logic is converted to a pixel stream at 1 pixel per clock in the CSI module. This pixel stream output is further processed by other modules (ISP or the rest of VI datapath or EPP). Depending on where the CSI pixel stream is processed (ISP, VI, or EPP), the maximum pixel clock frequency is limited by the pixel clock frequency of the modules that process this pixel stream.

> **Note:** VI/ISP/EPP pixel frequency may be lower than CSI output pixel clock frequency and therefore, may limit the actual pixel data rate. Please refer to the ISP, VI and EPP specifications.

Note that with multiple data lanes per CSI interface, the maximum serial data link speed may not be achievable due to pixel clock frequency limitations. Also, if two data streams come from the same CSI interface, since the streams are interleaved in time, the pixel clock frequency limitations may also limit the frame rate of the combined streams.

Table 104: Packet Efficiency

	Best* WC=65536	Typical* WC=1024
Payload	524 µs	8192 ns
Overhead	356 ns	356 ns
Efficiency	99.9%	96%

30.7 Error Resilience

The CSI is required to tolerate and/or recover stream errors at various levels.

According to the MIPI CSI 2.0 specification, the CSI must be able to detect and correct one-bit header error, detect two or more bits header errors, and detect possible payload errors using 16-bit CRC.

Due to the architectural constraints in VI and EPP, the CSI has to always send “valid” stream data to VI, even when the incoming stream contains errors and causes the CSI to drop packets. For stream data with a format other than *embedded*, “valid” stream data means one or more full frames of data that VI exactly expects for certain frame width and frame height.

If the incoming stream data (non-embedded) contains less bytes than the word count (WC), or less number of lines than the expected frame height, CSI must be able to pad pixels to short lines, or pad lines to short frames, so that CSI always sends “valid” frame data to VI.

30.8 Other Architectural Constraints

The following specifies other architectural constraints for this design:

- Because there are only 2 pixel parsers, embedded data and virtual channel cannot be supported simultaneously.
- Embedded data in a frame can be output in the same output port as the pixel data stream; however, in some cases where image processing or data reformatting is performed in VI/ISP/EPP, this embedded data may be accidentally processed in VI/ISP/EPP. Currently, VI/ISP/EPP cannot differentiate between embedded data or pixel data.
- Same stream going to different pixel parsers. This case will work as long as no stall comes from VI side. If a stall comes, then the current frame will be invalid. This is an exception case which should never normally occur. STM_ERR will be generated and software needs to take appropriate action.

30.9 CSI Datapath Module

The CSI datapath consists of three input ports and two output ports. There are two datapaths for pixel stream processing; therefore, at any time, a maximum of two input ports may be active simultaneously.

The components of the CSI datapath include:

- Three input ports from: CSI A interface, CSI B interface, and 32-bit host interface. Each port is capable of carrying a stream with CSI compatible data format.
- Two asynchronous input buffers (FIFOs) to receive streams from the CSI A interface and the CSI B interface.
- Three header parsers for searching packet header and to perform error detection and correction of CSI header.
- Two pixel parsers with corresponding output ports. Each pixel parser can convert a CSI packet data stream to output a pixel stream. Each output port consists of a maximum 24 bits of data per clock. Depending on the output data format options, one or two pixels per clock may be sent.

30.9.1 Header Parser

CSI pixel stream processing mainly consists of header parser and pixel parser. There are 3 header parsers: header parser A, header parser B, and header parser H.

Header parser A is enabled when CSI A interface is selected as input source. Similarly, header parser B is enabled when CSI B interface is selected as input source.

In general the header parser is used for:

- Detecting long and short packet headers and deciding what to do with the packet. This includes performing error detection and correction on the packet header prior to making decision and dealing with uncorrectable packet header.
- Skipping extra data on longer than expected data packets.
- Skipping next packet on imminent input buffer overflow when the pixel parser is busy processing current packet.

When enabled, the header parser will interpret CSI packet header, perform error detection and correction. If the packet header is uncorrectable, then an error is flagged and interrupt generated. If good or correctable CSI packet header is found, then the header parser will check the Data Identifier (DI) byte in the packet header and check the 2-bit Virtual Channel (VC) identifier and the 6-bit Data Type (DT) within this Data Identifier byte and will also check the 2-byte Word Count (WC) value in the packet header. If the virtual channel ID does not match the expected (programmed) value for the corresponding pixel parser then the packet is discarded. If the virtual channel ID matches the expected (programmed) value for the corresponding pixel parser then the header parser must decide what to do with the packet depending on the Data Type and the Word Count value. For long packet, if the Data Type does not match the expected (programmed) value for the corresponding pixel parser then the packet is discarded. If the Data Type of the long packet matches the expected (programmed) value for the corresponding pixel parser, then the corresponding pixel parser is notified to process the remaining packet data and checksum.

When first enabled, header parser must always search for Frame Start packet which is a CSI short packet. However, when input source is the host, there is an option to indicate Frame Start using the incoming vertical sync signal. If the vertical sync signal is used to indicate frame start and the CSI frame start packet is encountered, then the header parser will discard it. When frame start packet/signal is found, the header parser will notify the corresponding pixel parser so that it can prepare to process a new frame and output frame start code.

30.9.1.1 Data Type

There are 64 possible data types based on the 6-bit Data Type value.

Data Type	Description
0x00 – 0x07	Synchronization Short Packet Data Types
0x08 – 0x0F	Generic Short Packet Data Types
0x10 – 0x17	Generic Long Packet Data Types
0x18 – 0x1F	YUV Data
0x20 – 0x27	RGB Data
0x28 – 0x2F	RAW Data
0x30 – 0x37	User Defined Byte-based Data
0x38 – 0x3F	Reserved

30.9.1.2 Short Packets

There are 16 possible short packets based on Data Type value.

Data Type	Description
0x00	Frame Start Code
0x01	Frame End Code
0x02	Line Start Code (Optional)
0x03	Line End Code (Optional)
0x04 – 0x07	Reserved

30.9.1.3 Long Packets

CSI long packets are data packets which always consist of 1 line of data per packet with exception of arbitrary data packets. There are various data types defined. Please refer to the Input Data Format section for a summary of CSI data formats and their corresponding CSI module internal output format.

Generic long packets need special handling. There are 3 generic long packet types defined: null packet (DT=0x10), blanking data packet (DT=0x11), and embedded data packet (DT=0x12). There are also 5 reserved generic long packet types (DT=0x13 to 0x17). The header parser should discard all null packets and all reserved generic long packets.

Blanking data packets may contain blank color or blank image information. The transmission of blanking data packet is optional in the CSI2.0 specification. Null and blanking data are defined in the CSI2.0 specification mainly to accommodate a receiver which is timing sensitive and requires line start/end for every line in the frame. As a CSI receiver, this product does not have such blank timing sensitivity and it typically does not process blank data and does not store it to memory. Blanking data packet can therefore be discarded by the header parser.

However, some module may have a requirement for specific number of “blank” or extra lines at the end of vertical active area, such as the ISP module, which requires input active scan area to be about 6 lines larger than output active area. If a module that takes CSI output stream requires extra lines at the end of active image then blanking packet data can be used to generate these additional lines. In the future, there might be other reasons to process blank data. So, an option should be provided to accept and process blanking data. In the absence of better definition of blanking data format in the CSI specification, if software decides that blanking data cannot be discarded then, the pixel parser should assume that the blanking data format is the same as the programmed expected data type of the pixel stream.

The current CSI2.0 specification specifies that embedded data packets consists of 8-bit arbitrary data. This embedded data is, typically not the same pixel color as the image data. It might be used to send headers/status for JPEG/MPEG compressed data at the beginning or end of image data or in between image data lines. It might also be used to send closed caption text information or other type of information.

The exact use of embedded data is not clear, and therefore in most cases, embedded data is probably not used, or if sent by the transmitter, the embedded data packets can simply be discarded. If for some reason, it is important to receive the embedded data, there are other options that should be supported. Since there are two pixel parsers, if there is only one video source, it is best to direct embedded data to the other pixel parser which does not process the normal image data. In this way, the embedded data can be treated the same way as 8-bit arbitrary/compressed image data and can be output in 8-bit/clock or 16-bit/clock format. If there are two video sources that occupy both the pixel parsers, then embedded data if it has to be stored to memory, needs to be output in the same channel as the image data as 8-bit/16-bit data. But this also means that the module that receives output of CSI module and does image processing must have the ability to differentiate embedded data and not do image processing on this data prior to writing it to memory.

30.9.1.4 Top or Bottom Field Tag

During frame start, a tag is passed from CSI to VI to indicate top or bottom field. The 1-bit tag is at the least significant bit of the CSI output bus. If the bit is “1”, the field is top, but bottom otherwise.

30.10 Test Pattern Generator (TPG)

The test pattern generator is a configurable resource introduced to improve hardware verification capability for the Tegra CSI. Because a provision for inserting pixel data from the Host interface is not available, the hardware verification of the CSI and VI unit is difficult. The only sources of pixel data that can exercise the CSI are image sensors. In general, the sensor data cannot be controlled, making any self-checking test impossible. Test pattern generation modes offered by sensors are also ineffective because they have limited control.

There are two separate test pattern generators that can be configured to provide for the generation of synthetic image data, which is delivered to the PPA and PPB input FIFOs. The image data is multiplexed into the CSI data patch between lane-merging logic and the data FIFOs. Programmable configuration parameters are available to allow variation in pixel data, data-type, and spacing between various packets. In the real system, the LP state transition between any two CSI packets guarantees some idle time between the lines.

A difference engine architecture is employed which can generate almost arbitrary patterns that consist of frequency sweeps or intensity ramps in both the horizontal and vertical directions. In addition, by using selected bits from the difference engine variables as addresses, a color patchwork pattern can be generated. The output from the difference engine is modeled off of a second-order partial differential equation of the form:

$$\varphi_{out} = \varphi + x \cdot \frac{d\varphi}{dx} + x^2 \cdot \frac{d^2\varphi}{dx^2}$$

Where:

$$\varphi = PHASE$$

$$\frac{d\varphi}{dx} = FREQ$$

$$\frac{d^2\varphi}{dx^2} = FREQ\_RATE$$

The PHASE, FREQ, and FREQ_RATE variables may be programmed to create a variety of different test patterns and are either used directly or as an index into a look-up table (LUT). Table 105 summarizes the configuration parameters for the TPG.

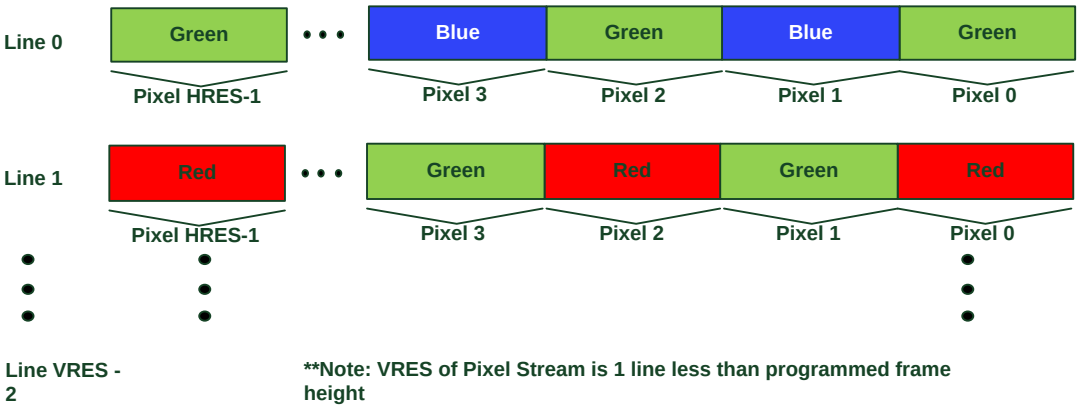
Table 105: Description of TPG Configuration Parameters

Configuration Variables for CSI_0 and CSI_1 Test Pattern Generators	TPG Frame Configuration	Image Height	Lines per frame
		Image Width	Pixels per line
		Vertical Blanking	Blank lines between frames
		Horizontal Blanking	Blank pixels between lines
		Initial Phase	Initial phase
		Mode	Direct or color patch
		Format	RAW8 support
	TPG Channel Configurations (R, G, and B)	Vertical Initial Frequency	Initial frequency
		Horizontal Initial Frequency	Initial frequency
		Vertical Frequency Rate	Rate of change for vertical frequency
		Horizontal Frequency Rate	Rate of change for horizontal frequency

30.10.1 TPG Frame Configuration Parameters

The TPG can be configured to provide the test images in Bayer RAW8 data format only. For the Bayer RAW8 data, the color components of the RGB pixel are sampled in an alternating fashion, depending on whether the current row and pixel within the row is even or odd.

Figure 124: RAW8 TPG Pixel Streams



The data from the test pattern generator has the same format (with the packet header, CRC, ECC, etc.) as received on the output of lane-merging logic during normal operation.

30.10.2 Modes of Operation

The test pattern generator operates in two modes:

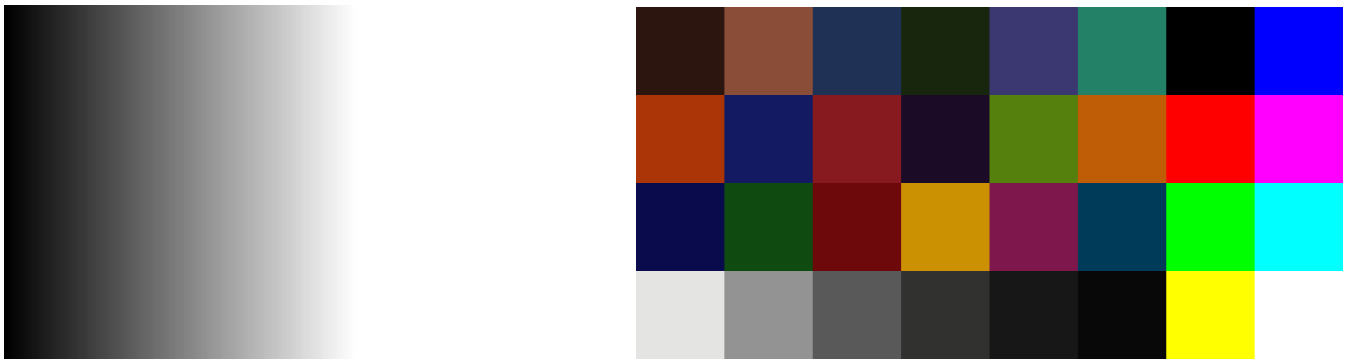
- Direct — The output of the difference engine computation will be viewed directly.
- Color Patch — The difference engine output will be used to generate a patchwork of colors.

In Direct mode, the output comes directly from the value of the phase accumulator in the difference engine. An example of the kind of pattern that can be made is shown in Figure 125(a). In Color Patch mode, the output comes from indexing a 32-element LUT.

The various bits from both the horizontal and vertical phase accumulators are used as address bits into a LUT of standard color values, allowing a patchwork of color test patterns to be generated. The colors in the pattern consist of linear RGB space

representations of the 24 “Macbeth” test chart colors, shown on the left side of Figure 125(b), and the 8 “100% saturation” colors of a TV test pattern, on the right side of Figure 125(a). This gives a total of 32 color patches.

Figure 125: Example of (a) Direct Mode (b) Color Patch Generator Output



(a) (b)

30.11 Software Requirements

30.11.1 Error Counters

To help debug, three 32-bit error counters have been implemented. Software can program each counter to increment at the event of any one of many error conditions or status change. The error conditions are:

- Header errors corrected
- Header uncorrectable errors
- CRC errors
- Packets too short, actual length less than WC
- Overflow errors due to padding packets too short
- FIFO overflow errors
- Illegal word count
- SoT single-bit error
- SoT multi-bit error, packet discarded
- EoT sequence error
- ESC-entry error
- LP-CTRL error
- The statistics that can be monitored include:
- Line packets processed
- Total packets processed
- ESC command processed

30.11.2 Programming Sequence

The following sequence is recommended for capturing a single frame:

1. Set up CSI registers for use case such as number of lanes, virtual channel, etc.
2. Initialize and power up CSI interface

3. Wait for initialization time or done signal from calibration logic
4. Power up camera through the I^2C interface
5. All CSI data and clock lanes are in stop state, LP11
6. Initiate frame capture through the I^2C
7. Frame done, CSI goes back to stop state, LP11

30.11.3 Escape Mode Handling

In the MIPI PHY specification, an escape mode is available for low speed command and data transfer. In our implementation, the escape mode entry is detected inside CSICIL. The command byte will be deposited in a register and interrupt generated from CSI. Software reads the register and decodes. CSI supports only one escape mode command which is Ultra-Low Power Mode (ULPM).

ULPM Command

Upon receiving the ULPM, software will synchronously power down the CSI interface and the camera as follows:

1. Software puts the camera in sleep mode through I^2C
2. The camera sends a ULPM command to CSI
3. CSI receives the ULPM command and raises an interrupt
4. Software powers down CSI, which goes to sleep in sync with the camera's CSI transmitter

30.12 DPHY Modes of Operation

The MIPI DPHY has 3 basic modes of operation. High speed mode employs differential signaling and achieves data rate of 1 Gbps. Both the driver and receiver are matched to 100 ohms differential. The driver is voltage mode for lower power consumption as opposed to current mode.

Low power control mode employs single-ended CMOS signaling for handshaking between camera and host. In this mode, there is no clock and no maximum symbol time defined in the specification. The receiver samples the input using both edges of the internal pixel clock, so the timing resolution is half the clock period of the pixel clock period.

The low power escape mode employs self-clocking using both data P and N pins. The clock is generated by XOR of data P and N. The maximum transfer rate is 20 Mbps for a 50 ns bit time. In our implementation, the receiver samples the input using both edges of the internal pixel clock also. It is the goal to sample at least twice during the bit interval; therefore, the internal pixel clock should have a minimum frequency of 20 MHz.

Table 106: MIPI DPHY Mode of Operations

Modes	Description	Clock
High speed (HS)	High speed differential signaling. Up to 1 Gbps. Burst transmission for low power.	500 MHz differential
Low Power (LP) Control	Single-ended 1.2V CMOS level. Low speed signaling for handshaking. Supports ULPM sleep state.	No Clock
Low Power (LP) Escape	Same as above. Low speed signaling for data, used for escape command entry only. 20Mbps	2 stage synchronizers to VI clock

30.13 MIPI-CSI Registers

Refer to "Reading Register Tables" in the Introduction section for the register table protocol as well as recommendations for accessing registers.

30.13.1 CSI_VI_INPUT_STREAM_CONTROL_0

VI Input Stream Control

Offset: 0x200 | Byte Offset: 0x800 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
7	X	VIP_SF_GEN: VIP Start Frame Generation. Do not use vi2csi_vip_vsync to generate start frame (SF) or end frame (EF) markers in the pixel parser output stream. 0 = VSYNC_SF: Pulses on vi2csi_vip_vsync will be used to generate start frame (SF) and end frame (EF) markers in the pixel parser output stream. In Tegra 4 devices, only payload_only mode is supported in the VIP input stream path, and this field may always be programmed to VSYNC_SF. 1 = NO_VSYNC_SF

30.13.2 CSI_HOST_INPUT_STREAM_CONTROL_0

Host Input Stream Control

Offset: 0x202 | Byte Offset: 0x808 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
28:16	RW	X	HOST_FRAME_HEIGHT: Host Frame Height. Specifies the height of the host frame when the host is supplying CSI format payload only data to one of the CSI pixel parsers. Programmed Value = number of lines – 1
8	RO	X	HOST_END_OF_PACKET: Writing this bit with a 1 indicates End of Packet, when CSI Host data is being received in Packet Format. In Packet Format vi2csi_host_hsync is not used to indicate beginning of packet.
7	RW	X	HOST_SF_GEN: Host Start Frame Generation. Do not use CSI Host Line counter to generate start, or End, of Frame control outputs. This setting should only be used if HOST_DATA_FORMAT is set to PACKETS, and the Host data stream has frame sync packets. 0 = LINE_COUNTER: CSI Host Line counter will be used to generate Frame start and end control. To signal the start of the first frame, the pixel parser will send an SF control, and signal start of frame mark, when it is first enabled with Host as its source. This setting should be used when HOST_DATA_FORMAT is set to PAYLOAD_ONLY. 1 = SHORT_PACKETS
3:0	RW	X	HOST_DATA_FORMAT: Host Data Format. Data written to Y_FIFO_WRITE port should be in CSI packet format. To indicate end of packet, a 1 should be written to HOST_END_OF_PACKET. A 1 should also be written to HOST_END_OF_PACKET before writing the first word of packet data to Y_FIFO_WRITE. 0 = PAYLOAD_ONLY: Data written to Y_FIFO_WRITE port should be CSI line payload data only (no header, no footer, and no short packets). A value of 1 should not be written to HOST_END_OF_PACKET (end of packet pulse only gets generated when a 1 is written to this bit). First line will be indicated when one of the pixel parsers is first enabled with its CSI_PPA/B_STREAM_SOURCE set to "HOST". The values in the following PIXEL_STREAM_A/B_CONTROL0 fields, for the pixel parser that is receiving host data, will be ignored; CSI_PPA/B_PACKET_HEADER overridden with "NOT_SENT", CSI_PPA/B_DATA_IDENTIFIER overridden with "DISABLED", CSI_PPA/B_WORD_COUNT_SELECT overridden with "REGISTER". CSI_PPA/B_CRC_CHECK overridden with "DISABLE", CSI_PPA/B_VIRTUAL_CHANNEL_ID, CSI_PPA/B_EMBEDDED_DATA_OPTIONS, and CSI_PPA/B_HEADER_EC_ENABLE. CSI_PPA/B_DATA_TYPE should be programmed with the 6-bit data type that is to be used to interpret the stream. CSI_PPA/B_WORD_COUNT should be programmed with the number of bytes per line. 1 = PACKETS

30.13.3 CSI_INPUT_STREAM_A_CONTROL_0

CSI Input Stream A Control

Offset: 0x204 | Byte Offset: 0x810 | Read/Write: R/W | Reset: 0bxxxxxxxx011111111xxxxxxxxxxx0x101

Bit	Reset	Description
23:16	0xff	CSI_A_SKIP_PACKET_THRESHOLD: CSI-A Skip Packet Threshold. This value is compared against the internal FIFO that buffer the input streams. A packet will be skipped (discarded) if the pixel stream processor is busy (probably due to padding process of a short line) and the number of entries in the internal FIFO exceeds this threshold value. Note that each entry in the internal FIFO buffer is four bytes. To turn off this feature, set the value to its maximum value (all ones).
4	0x0	CSI_A_SKIP_PACKET_THRESHOLD_ENABLE: Enables skip packet threshold feature. Skip packet feature is enabled. 0 = DISABLE: Skip packet feature is disabled. 1 = ENABLE
2	0x1	CSI_A_BYPASS_ALIGN: Bypass aligning CSIA and CSIB lanes if the valids for the lanes are out of sync by one cycle
1:0	0x1	CSI_A_DATA_LANE: CSI-A Data Lane. 0= 1 data lane; 1= 2 data lanes; 2= 3 data lanes; 3= 4 data lanes

30.13.4 CSI_PIXEL_STREAM_A_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 0x206 | Byte Offset: 0x818 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000

Bit	Reset	Description
29:28	X	CSI_PPA_PAD_FRAME: CSI Pixel Parser A Pad Frame. This specifies how to deal with frames that are shorter (fewer lines) than expected. Short frames are usually caused by line packets being dropped because of packet errors. Expected frame height is specified in PPA_EXP_FRAME_HEIGHT. To do padding the value in CSI_PPA_WORD_COUNT needs to be set to the number of input bytes in each line's payload. Short frames will not be padded out. 0 = PAD0S: Lines of all zeros will be used to pad out frames that are shorter than expected height. PPA_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD0S. 1 = PAD1S: Lines of all ones will be used to pad out frames that are shorter than expected height. PPA_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this fields is set to PAD1S. 2 = NOPAD
27	X	CSI_PPA_HEADER_EC_ENABLE: CSI Pixel Parser A Packet Header Error Correction Enable. This parameter specifies whether single bit errors in the packet header will be automatically corrected, or not. Single bit errors in the header will not be corrected. Header ECC check will still set header ECC status bits and the packet will be processed by Pixel Parser A. DISABLE should not be used when processing interleaved streams (Same stream going to both PPA and PPB). 0 = ENABLE: Single bit errors in the header will be automatically corrected. 1 = DISABLE

Bit	Reset	Description
25:24	X	CSI_PPA_PAD_SHORT_LINE: CSI Pixel Parser A Pad Short Line. This specifies how to deal with shorter than expected line (the number of bytes received is less than the specified word count). A short line is not padded (will output less pixels than expected). This option is not recommended and may cause other modules that receives CSI output stream to hang up. 0 = PAD0S: short line is padded by pixel of zeros such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet may be discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 1 = PAD1S: short line is padded by pixel of ones such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet may be discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 2 = NOPAD
21:20	X	CSI_PPA_EMBEDDED_DATA_OPTIONS: CSI Pixel Parser A Embedded Data Options. This specifies how to deal with embedded data within the specified input stream assuming that the CSI_PPA_DATA_TYPE is not embedded data and assuming that embedded data is not already processed by other CSI pixel stream processor. Output embedded data as 8-bpp arbitrary data stream. 0 = DISCARD: discard (throw away) embedded data 1 = EMBEDDED
19:16	X	CSI_PPA_OUTPUT_FORMAT_OPTIONS: CSI Pixel Parser A Output Format Options. This parameter specifies options for output data format. Output for storing RAW data to memory through ISP. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 0 = ARBITRARY: Output as 8-bit arbitrary data stream This may be used for compressed JPEG stream 1 = PIXEL: Output the normal 1 pixel/clock. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 2 = PIXEL_REP: Same as PIXEL except MS color bits, for RGB_666, RGB_565, RGB_555, and RGB_444, will be replicated to their undefined LS bits. 3 = STORE
15:14	X	CSI_PPA_VIRTUAL_CHANNEL_ID: CSI Pixel Parser A Virtual Channel Identifier. This is CSI compatible virtual channel identifier as defined in CSI specification. If the source stream contains packet headers and CSI_PPA_DATA_IDENTIFIER is ENABLED this value will be compared to the CSI Virtual Channel Identifier value in the 2 MSBs of the CSI Data Identifier (DI) byte. This value will be ignored if the source stream does not contain packet headers, or CSI_PPA_DATA_IDENTIFIER is DISABLED, then this value will be ignored. 0 = ONE 1 = TWO 2 = THREE 3 = FOUR
13:8	X	CSI_PPA_DATA_TYPE: CSI Pixel Parser A Data Type. This is CSI compatible data type as defined in CSI specification. If the source stream contains packet headers this value can be compared to the CSI Data Type value in the 6 LSBs of the CSI Data Identifier (DI) byte. If the source stream does not contain packet headers, or CSI_PPA_DATA_IDENTIFIER is DISABLED, this value will be used to determine how the stream will be converted to pixels. 24 = YUV420_8 25 = YUV420_10 26 = LEG_YUV420_8 28 = YUV420CSPS_8 29 = YUV420CSPS_10 30 = YUV422_8 31 = YUV422_10 32 = RGB444 33 = RGB555 34 = RGB565 35 = RGB666 36 = RGB888 40 = RAW6 41 = RAW7 42 = RAW8 43 = RAW10 44 = RAW12

Bit	Reset	Description
		45 = RAW14 48 = ARB_DT1 49 = ARB_DT2 50 = ARB_DT3 51 = ARB_DT4
7	X	CSI_PPA_CRC_CHECK: CSI Pixel Parser A Data CRC Check. This parameter specifies whether the last 2 bytes of packet should be treated as CRC checksum and used to perform CRC check on the payload data. Note that in case there are 2 bytes of data CRC at the end of the packet, the packet word count does not include the CRC bytes. Data CRC Check is enabled. 0 = DISABLE: Data CRC Check is disabled regardless of whether there are CRC checksum at the end of the packet. 1 = ENABLE
6	X	CSI_PPA_WORD_COUNT_SELECT: CSI Pixel Parser A Word Count Select. This parameter is effective only if packet header is sent as part of the stream. The number of bytes per line is to be extracted from Word Count field in the packet header. Note that if the serial link is not error free, programming this bit to HEADER may be dangerous because the word count information in the header may be corrupted. 0 = REGISTER: Word Count in packet header is ignored and the number of bytes per line/packet is specified by CSI_PPA_WORD_COUNT. Payload CRC check will not be valid if the word count in CSI_PPA_WORD_COUNT is different than the count in the packet header. It is recommended to always program this bit to REGISTER and always program CSI_PPA_WORD_COUNT. 1 = HEADER
5	X	CSI_PPA_DATA_IDENTIFIER: CSI Pixel Parser A Data Identifier (DI) byte processing. This parameter is effective only if packet header is sent as part of the stream. Enabled - Data Identifier byte in packet header should be compared against the CSI_PPA_DATA_TYPE and the CSI_PPA_VIRTUAL_CHANNEL_ID. 0 = DISABLED: Disabled - Data Identifier byte in packet header should be ignored (not checked against CSI_PPA_DATA_TYPE and against CSI_PPA_VIRTUAL_CHANNEL_ID). In this case, CSI_PPA_DATA_TYPE specifies the stream data format. 1 = ENABLED
4	X	CSI_PPA_PACKET_HEADER: CSI Pixel Parser A Packet Header processing. This specifies whether packet header is sent in the beginning of packet or not. Packet header is sent. This setting should be used if the stream source is CSI Interface A or B. 0 = NOT_SENT: Packet header is not sent. This setting should not be used if the stream source is CSI Interface A or B. Unless CSI-A, or CSI-B, is operating in a stream capture debug mode. In this case, CSI_PPA_DATA_TYPE specifies the stream data format and the number of bytes per line/packet is specified by CSI_PPA_WORD_COUNT. This implies that a packet footer is also not sent. In this case, no packet footer CRC check should be performed. 1 = SENT
2:0	0x0	CSI_PPA_STREAM_SOURCE: CSI Pixel Parser A Stream Source Host 0 = CSI_A: CSI Interface A 1 = CSI_B: CSI Interface B 6 = VI_PORT: VI port 7 = HOST

30.13.5 CSI_PIXEL_STREAM_A_CONTROL1_0

CSI Pixel Stream A Control 1

Offset: 0x207 | Byte Offset: 0x81c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000|

Bit	Reset	Description
7:4	0x0	CSI_PPA_TOP_FIELD_FRAME_MASK: CSI Pixel Parser A Top Field Frame Mask
3:0	0x0	CSI_PPA_TOP_FIELD_FRAME: CSI Pixel Parser A Top Field Frame. This parameter specifies the frame number for top field detection for interlaced input video stream. Top Field is indicated when each of the least significant four bits of the frame number that has a one in its mask bit matches the corresponding bit in this parameter. In other words, Top

Bit	Reset	Description
		Field is detected when the bitwise AND of ~(CSI_PPA_TOP_FIELD_FRAME ^ <frame number>) & CSI_PPA_TOP_FIELD_FRAME_MASK is one. Frame Number is taken from the WC field of the Frame Start short packet.

30.13.6 CSI_PIXEL_STREAM_A_WORD_COUNT_0

CSI Pixel Stream A Word Count

Offset: 0x208 | Byte Offset: 0x820 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx |

Bit	Reset	Description
15:0	X	CSI_PPA_WORD_COUNT: CSI Pixel Parser A Word Count. This parameter specifies the number of bytes per line/packet in the case where Word Count field in packet header is not used or where packet header is not sent. This count does not include the additional 2 bytes of CRC checksum if data CRC check is enabled. When the input stream comes from a CSI camera port, this parameter must be programmed when CSI_PPA_PAD_SHORT_LINE is set to either PAD0S or PAD1S, no matter whether CSI_PPA_WORD_COUNT_SELECT is set to REGISTER or HEADER. When the input stream comes from the host path or from the VIP path, and the data mode is PAYLOAD_ONLY, this count must be programmed. Given a line width of N pixels, the programming value of this parameters is as follows **Data format** — **Value** YUV420_8 — N bytes YUV420_10 — N/4*5 bytes LEG_YUV420_8 — N/2*3 bytes YUV422_8 — N*2 bytes YUV422_10 — N/2*5 bytes RGB888 — N*3 bytes RGB666 — N/4*9 bytes RGB565 — N*2 bytes RGB555 — N*2 bytes RGB444 — N*2 bytes RAW6 — N/4*3 bytes RAW7 — N/8*7 bytes RAW8 — N bytes RAW10 — N/4*5 bytes RAW12 — N/2*3 bytes RAW14 — N/4*7 bytes

30.13.7 CSI_PIXEL_STREAM_A_GAP_0

CSI Pixel Stream A Gap

Offset: 0x209 | Byte Offset: 0x824 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	PPA_FRAME_MIN_GAP: Minimum number of viclk cycles from end of frame (Video_control = EF) to start of next frame (Video_control = SF). This parameter ensures that the minimum V-blank time requirement of VI/ISP is satisfied. This field takes effect only when the frame gap of the input stream is less than the specified value.
15:0	0x0	PPA_LINE_MIN_GAP: Minimum number of viclk cycles from end of previous line (Video_control = EL_DATA) to start of next line (Video_control = SL). This parameter ensures that the minimum H-blank time requirement of VI/ISP is satisfied. This field takes effect only when the line gap of the input stream is less than the specified value.

30.13.8 CSI_PIXEL_STREAM_PPA_COMMAND_0

CSI Pixel Parser A Command

Offset: 0x20a | Byte Offset: 0x828 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
15:12	X	CSI_PPA_START_MARKER_FRAME_MAX: CSI Pixel Parser A Start Marker Maximum. Start Frame is indicated when the Minimum condition above is met and the least significant four bits of the frame number are less than or equal to this value.
11:8	X	CSI_PPA_START_MARKER_FRAME_MIN: CSI Pixel Parser A Start Marker Minimum. Start Frame is indicated when the Maximum condition below is met and the least significant four bits of the frame number are greater than or equal to this value.
4	X	CSI_PPA_VSYNC_START_MARKER: CSI Pixel Parser A VSYNC Start Marker. The start of frame is indicated when the VSYNC signal is received. When the input stream is from the VIP path and the data mode is PACKET, then this field may be programmed to VSYNC. 0 = FSPKT: Start of frame is indicated when a Frame Start short packet is received with a frame number whose least significant four bits are greater than or equal to CSI_PPA_START_MARKER_FRAME_MIN and less than or equal to CSI_PPA_START_MARKER_FRAME_MAX. When the input stream is from a CSI port, or from the host path, or from the VIP path and the data mode is PAYLOAD_ONLY, then this field may be programmed to FSPKT. 1 = VSYNC
2	X	CSI_PPA_SINGLE_SHOT: CSI Pixel Parser A Single Shot Mode. Software should clear it along with disabling the CSI_PPA_ENABLE, once a frame is captured 0 = DISABLE 1 = ENABLE
1:0	0x0	CSI_PPA_ENABLE: CSI Pixel Parser A Enable. This parameter controls CSI Pixel Parser A to start or stop receiving data. Reset (disable immediately) Enabling the pixel parser does not enable the corresponding input source to receive data. If the pixel parser is enabled later than the corresponding input source, the CSI will keep on rejecting incoming stream, until it encounters a valid SF. 0 = NOP: no operation 1 = ENABLE: enable at the next frame start as specified by the CSI Start Marker 2 = DISABLE: disable after current frame end and before next frame start. 3 = RST

30.13.9 CSI_PIXEL_STREAM_A_EXPECTED_FRAME_0

CSI Pixel Stream A Expected Frame

Offset: 0x232 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	PPA_EXP_FRAME_HEIGHT: CSI-PPA Expected Frame Height. Specifies the expected height of the CSI-PPA frame output. Padding out of frames that are shorter than this expected height can be specified using CSI_PPA_PAD_FRAME. If CSI_PPA_PAD_FRAME is set to PAD0S or PAD1S, this parameter must be programmed. If CSI_PPA_PAD_FRAME is set to NOPAD, this parameter may not be programmed. Programmed Value = number of lines
15:4	X	PPA_MAX_CLOCKS: Maximum Number of viclk clock cycles between line start requests. The value in this field is in terms of 256 viclk clock cycles.
0	X	PPA_ENABLE_LINE_TIMEOUT: When set to one, enables checking of the time between start line requests from the Header Parser to CSI-PPA. A fake EF will be output by CSI-PPA if this time between line starts exceeds the value in MAX_CLOCKS_BETWEEN_LINES. Padding lines can be inserted before the fake EF, if the number of lines output, when the fake EF is generated is less than the expected frame height. The type of padding is specified using CSI_PPA_PAD_FRAME.

30.13.10 CSI_INPUT_STREAM_B_CONTROL_0

CSI Input Stream B Control

Offset: 0x20f | Byte Offset: 0x83c | Read/Write: R/W | Reset: 0bxxxxxxx011111111xxxxxxxxxx0x101

Bit	Reset	Description
24:16	0xff	CSI_B_SKIP_PACKET_THRESHOLD: CSI-B Skip Packet Threshold. This value is compared against the internal FIFO that buffers the input streams. A packet will be skipped (discarded) if the pixel stream processor is busy (probably due to padding process of a short line) and the number of entries in the internal FIFO exceeds this threshold value. Note that each entry in the internal FIFO buffer is four bytes. To turn off this feature, set the value to its maximum value (all ones).
4	0x0	CSI_B_SKIP_PACKET_THRESHOLD_ENABLE: Enables skip packet threshold feature. Skip packet feature is enabled. 0 = DISABLE: Skip packet feature is disabled. 1 = ENABLE
2	0x1	CSI_B_BYPASS_ALIGN: Bypass aligning CSIC and CSID lanes if the valids for the lanes are out of sync by one cycle
1:0	0x1	CSI_B_DATA_LANE: CSI-B Data Lane. 0= 1 data lane 1= 2 data lanes 2= 3 data lanes 3= 4 data lanes

30.13.11 CSI_PIXEL_STREAM_B_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 0x211 | Byte Offset: 0x844 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000

Bit	Reset	Description
29:28	X	CSI_PPB_PAD_FRAME: CSI Pixel Parser B Pad Frame. This specifies how to deal with frames that are shorter (fewer lines) than expected. Short frames are usually caused by line packets being dropped because of packet errors. Expected frame height is specified in PPB_EXP_FRAME_HEIGHT. To do padding the value in CSI_PPB_WORD_COUNT needs to be set to the number of input bytes in each lines payload. Short frames will not be padded out. 0 = PAD0S: Lines of all zeros will be used to pad out frames that are shorter than expected height. PPB_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD0S. 1 = PAD1S: Lines of all ones will be used to pad out frames that are shorter than expected height. PPB_EXP_FRAME_HEIGHT must be programmed to an appropriate value if this field is set to PAD1S. 2 = NOPAD
27	X	CSI_PPB_HEADER_EC_ENABLE: CSI Pixel Parser B Packet Header Error Correction Enable. This parameter specifies whether single bit errors in the packet header will be automatically corrected or not. Single bit errors in the header will not be corrected. Header ECC check will still set header ECC status bits and the packet will be processed by Pixel Parser B. DISABLE should not be used when processing interleaved streams (Same stream going to both PPA and PPB). 0 = ENABLE: Single bit errors in the header will be automatically corrected. 1 = DISABLE

Bit	Reset	Description
25:24	X	CSI_PPB_PAD_SHORT_LINE: CSI Pixel Parser B Pad Short Line. This specifies how to deal with shorter than expected line (the number of bytes received is less than the specified word count) short line is not padded (will output less pixels than expected). This option is not recommended and may cause other modules that receive CSI output stream to hang up. 0 = PAD0S: short line is padded by pixel of zeros such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet may be discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 1 = PAD1S: short line is padded by pixel of ones such that the expected number of output pixels is correct. Due to the time required to do the padding, subsequent line packet may be discarded and therefore may cause a short frame (total number of lines per frame is less than expected). 2 = NOPAD
21:20	X	CSI_PPB_EMBEDDED_DATA_OPTIONS: CSI Pixel Parser B Embedded Data Options. This specifies how to deal with embedded data within the specified input stream assuming that the CSI_PPB_DATA_TYPE is not embedded data and that embedded data is not already processed by another CSI pixel stream processor. Output embedded data as 8-bpp arbitrary data stream. 0 = DISCARD: discard (throw away) embedded data 1 = EMBEDDED
19:16	X	CSI_PPB_OUTPUT_FORMAT_OPTIONS: CSI Pixel Parser B Output Format Options. This parameter specifies output data format. Output for storing RAW data to memory through ISP. Undefined LS color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 0 = ARBITRARY: Output as 8-bit arbitrary data stream This may be used for compressed JPEG stream 1 = PIXEL: Output the normal 1 pixel/clock. Undefined LS. color bits for RGB_666, RGB_565, RGB_555, and RGB_444, will be zeroed. 2 = PIXEL_REP: Same as PIXEL except MS color bits, for RGB_666, RGB_565, RGB_555, and RGB_444, will be replicated to their undefined LS bits. 3 = STORE
15:14	X	CSI_PPB_VIRTUAL_CHANNEL_ID: CSI Pixel Parser B Virtual Channel Identifier. This is CSI compatible virtual channel identifier as defined in the CSI specification. If the source stream contains packet headers and CSI_PPB_DATA_IDENTIFIER is ENABLED this value will be compared to the CSI Virtual Channel Identifier value in the 2 MSBs of the CSI Data Identifier (DI) byte. This value will be ignored if the source stream does not contain packet headers, or CSI_PPB_DATA_IDENTIFIER is DISABLED, then this value will be ignored. 0 = ONE 1 = TWO 2 = THREE 3 = FOUR

Bit	Reset	Description
13:8	X	CSI_PPB_DATA_TYPE: CSI Pixel Parser B Data Type. This is CSI compatible data type as defined in CSI specification. If the source stream contains packet headers this value can be compared to the CSI Data Type value in the 6 LSBs of the CSI Data Identifier (DI) byte. If the source stream does not contain packet headers or CSI_PPB_DATA_IDENTIFIER is DISABLED, this value will be used to determine how the stream will be converted to pixels. 24 = YUV420_8 25 = YUV420_10 26 = LEG_YUV420_8 28 = YUV420CSPS_8 29 = YUV420CSPS_10 30 = YUV422_8 31 = YUV422_10 32 = RGB444 33 = RGB555 34 = RGB565 35 = RGB666 36 = RGB888 40 = RAW6 41 = RAW7 42 = RAW8 43 = RAW10 44 = RAW12 45 = RAW14 48 = ARB_DT1 49 = ARB_DT2 50 = ARB_DT3 51 = ARB_DT4
7	X	CSI_PPB_CRC_CHECK: CSI Pixel Parser B Data CRC Check. This parameter specifies whether the last 2 bytes of packet should be treated as CRC checksum and used to perform CRC check on the payload data. Note that in case there are 2 bytes of data CRC at the end of the packet, the packet word count does not include the CRC bytes. Data CRC Check is enabled. 0 = DISABLE: Data CRC Check is disabled regardless of whether there are CRC checksum at the end of the packet. 1 = ENABLE
6	X	CSI_PPB_WORD_COUNT_SELECT: CSI Pixel Parser B Word Count Select. This parameter is effective only if packet header is sent as part of the stream. The number of bytes per line is to be extracted from Word Count field in the packet header. Note that if the serial link is not error free, programming this bit to HEADER may be dangerous because the word count information in the header may be corrupted. 0 = REGISTER: Word Count in packet header is ignored and the number of bytes per line/packet is specified by CSI_PPB_WORD_COUNT. Payload CRC check will not be valid if the word count in CSI_PPB_WORD_COUNT is different than the count in the packet header. It is recommended to always program this bit to REGISTER and always program CSI_PPB_WORD_COUNT. 1 = HEADER
5	X	CSI_PPB_DATA_IDENTIFIER: CSI Pixel Parser B Data Identifier (DI) byte processing. This parameter is effective only if packet header is sent as part of the stream. Enabled - Data Identifier byte in packet header should be compared against the CSI_PPB_DATA_TYPE and the CSI_PPB_VIRTUAL_CHANNEL_ID. 0 = DISABLED: Disabled - Data Identifier byte in packet header should be ignored (not checked against CSI_PPB_DATA_TYPE and against CSI_PPB_VIRTUAL_CHANNEL_ID). In this case, CSI_PPB_DATA_TYPE specifies the stream data format. 1 = ENABLED

Bit	Reset	Description
4	X	CSI_PPB_PACKET_HEADER: CSI Pixel Parser B Packet Header processing. This specifies whether packet header is sent in the beginning of packet or not. Packet header is sent. This setting should be used if the stream source is CSI Interface A or B. 0 = NOT_SENT: Packet header is not sent. This setting should not be used if the stream source is CSI Interface A or B. Unless CSI-A, or CSI-B, is operating in a stream capture debug mode. In this case, CSI_PPB_DATA_TYPE specifies the stream data format and the number of bytes per line/packet is specified by CSI_PPB_WORD_COUNT. This implies that a packet footer is also not sent. In this case, no packet footer CRC check should be performed. 1 = SENT
2:0	0x0	CSI_PPB_STREAM_SOURCE: CSI Pixel Parser B Stream Source Host 0 = CSI_A: CSI Interface A 1 = CSI_B: CSI Interface B 6 = VI_PORT: VI port 7 = HOST

30.13.12 CSI_PIXEL_STREAM_B_CONTROL1_0

CSI Pixel Stream B Control 1

Offset: 0x212 | Byte Offset: 0x848 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
7:4	0x0	CSI_PPB_TOP_FIELD_FRAME_MASK: CSI Pixel Parser B Top Field Frame Mask
3:0	0x0	CSI_PPB_TOP_FIELD_FRAME: CSI Pixel Parser B Top Field Frame. This parameter specifies the frame number for top field detection for interlaced input video stream. Top Field is indicated when each of the least significant four bits of the frame number that has a one in its mask bit matches the corresponding bit in this parameter. In other words, Top Field is detected when the bitwise AND of ~(CSI_PPB_TOP_FIELD_FRAME ^ <frame number>) & CSI_PPB_TOP_FIELD_FRAME_MASK is one. Frame Number is taken from the WC field of the Frame Start short packet.

30.13.13 CSI_PIXEL_STREAM_B_WORD_COUNT_0

CSI Pixel Stream A Word Count

Offset: 0x213 | Byte Offset: 0x84c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	CSI_PPB_WORD_COUNT: CSI Pixel Parser B Word Count. This parameter specifies the number of bytes per line/packet in the case where Word Count field in packet header is not used or where packet header is not sent. This count does not include the additional 2 bytes of CRC checksum if data CRC check is enabled. When the input stream comes from a CSI camera port, this parameter must be programmed when CSI_PPB_PAD_SHORT_LINE is set to either PAD0S or PAD1S, no matter whether CSI_PPB_WORD_COUNT_SELECT is set to REGISTER or HEADER. When the input stream comes from the host path or from the VIP path, and the data mode is PAYLOAD_ONLY, this count must be programmed. Given a line width of N pixels, the programming value of this parameters is as follows: **Data Format** — **Value** YUV420_8 — N bytes YUV420_10 — N/4*5 bytes LEG_YUV420_8 — N/2*3 bytes YUV422_8 — N*2 bytes YUV422_10 — N/2*5 bytes RGB888 — N*3 bytes RGB666 — N/4*9 bytes RGB565 — N*2 bytes RGB555 — N*2 bytes

Bit	Reset	Description
		RGB444 N*2 bytes RAW6 N/4*3 bytes RAW7 N/8*7 bytes RAW8 N bytes RAW10 N/4*5 bytes RAW12 N/2*3 bytes RAW14 N/4*7 bytes

30.13.14 CSI_PIXEL_STREAM_B_GAP_0

CSI Pixel Stream B Gap

Offset: 0x214 | Byte Offset: 0x850 | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	PPB_FRAME_MIN_GAP: Minimum number of viclk cycles from end of frame (Video_control = EF) to start of next frame (Video_control = SF). This parameter ensures that the minimum V-blank time requirement of VI/ISP is satisfied. This field takes effect only when the frame gap of the input stream is less than the specified value.
15:0	0x0	PPB_LINE_MIN_GAP: Minimum number of viclk cycles from end of previous line (Video_control = EL_DATA) to start of next line (Video_control = SL). This parameter ensures that the minimum H-blank time requirement of VI/ISP is satisfied. This field takes effect only when the line gap of the input stream is less than the specified value.

30.13.15 CSI_PIXEL_STREAM_PPB_COMMAND_0

CSI Pixel Parser B Command

Offset: 0x215 | Byte Offset: 0x854 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
15:12	X	CSI_PPB_START_MARKER_FRAME_MAX: CSI Pixel Parser B Start Marker Maximum. Start Frame is indicated when the Minimum condition above is met and the least significant four bits of the frame number are less than, or equal to, this value.
11:8	X	CSI_PPB_START_MARKER_FRAME_MIN: CSI Pixel Parser B Start Marker Minimum. Start Frame is indicated when the Maximum condition below is met and the least significant four bits of the frame number are greater than, or equal to, this value.
4	X	CSI_PPB_VSYNC_START_MARKER: CSI Pixel Parser B VSYNC Start Marker. Start of frame is indicated when VSYNC signal is received. When the input stream is from the VIP path and the data mode is PACKET, then this field may be programmed to VSYNC. 0 = FSPKT: Start of frame is indicated when a Frame Start short packet is received with a frame number whose least significant four bits are greater than, or equal to, CSI_PPB_START_MARKER_FRAME_MIN and less than, or equal to, CSI_PPB_START_MARKER_FRAME_MAX. When the input stream is from a CSI port, or from the host path, or from the VIP path and the data mode is PAYLOAD_ONLY, then this field may be programmed to FSPKT. 1 = VSYNC
2	X	CSI_PPB_SINGLE_SHOT: CSI Pixel Parser B Single Shot Mode. Software should clear it along with disabling CSI_PPB_ENABLE, once a frame is captured 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1:0	0x0	CSI_PPB_ENABLE: CSI Pixel Parser B Enable. This parameter controls CSI Pixel Parser B to start or stop receiving data. reset (disable immediately). Enabling the pixel parser does not enable the corresponding input source to receive data. If the pixel parser is enabled later than the corresponding input source, the CSI will keep rejecting the incoming stream, until it encounters a valid SF. 0 = NOP: no operation 1 = ENABLE: enable at the next frame start as specified by the CSI Start Marker 2 = DISABLE: disable after current frame end and before next frame start. 3 = RST

30.13.16 CSI_PIXEL_STREAM_B_EXPECTED_FRAME_0

CSI Pixel Stream B Expected Frame

Offset: 0x233 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	PPB_EXP_FRAME_HEIGHT: CSI-PPB Expected Frame Height. Specifies the expected height of the CSI-PPB frame output. Padding out of frames that are shorter than this expected height can be specified using CSI_PPB_PAD_FRAME. If CSI_PPB_PAD_FRAME is set to PAD0S or PAD1S, this parameter must be programmed. If CSI_PPB_PAD_FRAME is set to NOPAD, this parameter may not be programmed. Programmed Value = number of lines
15:4	X	PPB_MAX_CLOCKS: Maximum Number of viclk clock cycles between line start requests. The value in this field is in terms of 256 viclk clock cycles.
0	X	PPB_ENABLE_LINE_TIMEOUT: When set to one, enables checking of the time between start line requests from the Header Parser to CSI-PPB. A fake EF will be output by CSI-PPB if this time between line starts exceeds the value in MAX_CLOCKS_BETWEEN_LINES. Padding lines can be inserted before the fake EF, if the number of lines output, when the fake EF is generated is less than the expected frame height. The type of padding is specified using CSI_PPB_PAD_FRAME.

30.13.17 CSI_PHY_CIL_COMMAND_0

CSI PHY and CIL Command

Offset: 0x21a | Byte Offset: 0x868 | Read/Write: R/W | Reset: 0bxx00xx00xx00xx00xxxxxxxxxxxxxx00

Bit	Reset	Description
29:28	0x0	CSI_E_PHY_CIL_ENABLE: CSI E PHY and CIL Enable. This parameter controls CSI E PHY and CIL receiver to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable 2 = DISABLE: reset
25:24	0x0	CSI_D_PHY_CIL_ENABLE: CSI D PHY and CIL Enable. This parameter controls CSI D PHY and CIL receiver to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable 2 = DISABLE: reset
21:20	0x0	CSI_C_PHY_CIL_ENABLE: CSI C PHY and CIL Enable. This parameter controls CSI C PHY and CIL receiver to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable 2 = DISABLE: reset

Bit	Reset	Description
17:16	0x0	CSI_B_PHY_CIL_ENABLE: CSI B PHY and CIL Enable. This parameter controls CSI B PHY and CIL receiver to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable 2 = DISABLE: reset
1:0	0x0	CSI_A_PHY_CIL_ENABLE: CSI A PHY and CIL Enable. This parameter controls CSI A PHY and CIL receiver to start or stop receiving data. 0 = NOP: no operation 1 = ENABLE: enable 2 = DISABLE: reset

30.13.18 CSI_PHY_CILA_CONTROL0_0

CSI-A PHY and CIL Control

Offset: 0x21b | Byte Offset: 0x86c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxx0000xx0x0010

Bit	Reset	Description
11:8	0x0	CILA_CLK_SETTLE: Settle time for clock lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (102 MHz LP clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 csicil cycles for the default value. So the programming value (0 to 15) of this field works like this: 0 = 15 cilclk cycles (default internal delay) 1 = 15 - 7 (8 cilclk cycles) 2 = 15 - 6 (9 cilclk cycles) … 7 = 15 - 1 (14 cilclk cycles) 8 = 15 - 0 (default internal delay) 9 = 15 + 1 (16 cilclk cycles) … 15 = 15 + 7 (22 cilclk cycles)
5	0x0	CILA_BYPASS_LP_SEQ: The LP signals should sequence through LP11->LP01->LP00 state, to indicate to CLOCK CIL about the mode switching to HS Rx mode. In case Camera is enabled earlier than CIL, it is highly likely that camera sends this control sequence sooner than CIL can detect it. Enabling this bit allows the CLOCK CIL to overlook the LP control sequence and step in HS Rx mode directly looking at LP00 only.
3:0	0x2	CILA_THS_SETTLE: Settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (102 MHz LP clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field. 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

30.13.19 CSI_PHY_CILB_CONTROL0_0

CSI-B PHY and CIL Control

Offset: 0x21c | Byte Offset: 0x870 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxx0000xx0x0010

Bit	Reset	Description
11:8	0x0	CILB_CLK_SETTLE: Settle time for clock lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (102 MHz LP clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 CSICIL cycles for default value. So the programming value (0 to 15) of this field works like this: 0 = 15 cilclk cycles (default internal delay) 1 = 15 - 7 (8 cilclk cycles) 2 = 15 - 6 (9 cilclk cycles) …

Bit	Reset	Description
		7 = 15 - 1 (14 cilclk cycles) 8 = 15 - 0 (default internal delay) 9 = 15 + 1 (16 cilclk cycles) … 15 = 15 + 7 (22 cilclk cycles)
5	0x0	CILB_BYPASS_LP_SEQ: see CILA_BYPASS_LP_SEQ above
3:0	0x2	CILB_THS_SETTLE: Settle time for data lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many CSICIL clock cycles (102 MHz LP clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field. 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

30.13.20 CSI_CSI_PIXEL_PARSER_STATUS_0

Pixel Parser Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x2 to CSI_PIXEL_PARSER_STATUS will clear only PPA_ILL_WD_CNT.

Offset: 0x21e | Byte Offset: 0x878 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	HPH_UNC_HDR_ERR: Uncorrectable Header Error. Set when the Host port header parser parses a header with a multi-bit error. This error will be detected by the header's ECC, but cannot be corrected. The packet will be discarded.
30	X	HPV_UNC_HDR_ERR: Uncorrectable Header Error. Set when the VI port header parser parses a header with a multi-bit error. This error will be detected by the header's ECC, but cannot be corrected. The packet will be discarded.
26	X	PPB_SPARE_STATUS_1: PPB Spare Status bit. This bit will get set when Pixel Parser B has a line timeout. Line timeout needs to be enabled by setting PPB_ENABLE_LINE_TIMEOUT and programming PPB_MAX_CLOCKS for the MAX clocks between lines.
25	X	PPB_INTERFRAME_LINE: Set when CSI-PPB receives a request to output a line that is not in the active part of the frame output. That is after EF and before SF, or before start marker is found. The interframe line will not be output by the Pixel Parser.
24	X	PPB_EXTRA_SF: Set when CSI-PPB receives an SF when it is expecting an EF. This happens when the EF of the frame gets corrupted before the arriving CSI. CSI-PPB will insert a fake EF and drop the current frame with Correct SF.
23	X	PPB_SHORT_FRAME: Set when CSI-PPB receives a short frame. This bit gets set even if CSI_PPB_PAD_FRAME specifies that short frames are to be padded to the correct line length.
22	X	PPB_STMERR: Stream Error. Set when the control output of PPB does not follow the correct sequence. The correct sequence for CSI is: SF -> (SL_DATA or EF), SL_DATA -> (DATA or EL_DATA), DATA -> EL_DATA, EL_DATA -> (SL_DATA or EF), EF -> SF. Stream Errors can be caused by receiving a corrupted stream.
21	X	PPB_FIFO_OVRF: FIFO Overflow. Set when the FIFO that is feeding packets to PPB overflows.
20	X	PPB_PL_CRC_ERR: PayLoad CRC Error. Set when a packet that was processed by PPB had a payload CRC error.
19	X	PPB_SL_PKT_DROPPED: Short Line Packet Dropped. Set when an incoming packet gets dropped because the input FIFO level reaches CSI_B_SKIP_PACKET_THRESHOLD when padding a short line.
18	X	PPB_SL_PROCESSED: Short Line Processed. Set when a line with a payload that is shorter than its packet header word count is processed by the PPB.

Bit	Reset	Description
17	X	PPB_ILL_WD_CNT: Illegal Word Count. Set when a line with a word count that does not generate an integer number of pixels (unused bytes at the end of payload) is processed by PPB.
16	X	PPB_HDR_ERR_COR: Header Error Corrected. Set when a packet that was processed by PPB has a single bit header error. This error will be detected by the header's ECC, and corrected by it if header error correction is enabled (CSI_B_HEADER_EC_ENABLE = 0). This flag will be set and the packet will be processed even if the error is not corrected.
15	X	HPB_UNC_HDR_ERR: Uncorrectable Header Error. Set when header parser B parses a header with a multi-bit error. This error will be detected by the header's ECC, but cannot be corrected. The packet will be discarded.
14	X	HPA_UNC_HDR_ERR: Uncorrectable Header Error. Set when header parser A parses a header with a multi bit error. This error will be detected by the headers ECC, but cannot be corrected. The packet will be discarded.
10	X	PPA_SPARE_STATUS_1: PPA Spare Status bit. This bit will get set when Pixel Parser A has a line timeout. Line timeout needs to be enabled by setting PPA_ENABLE_LINE_TIMEOUT and programming PPA_MAX_CLOCKS for the MAX clocks between lines.
9	X	PPA_INTERFRAME_LINE: Set when CSI-PPA receives a request to output a line that is not in the active part of the frame output. That is after EF and before SF, or before start marker is found. The interframe line will not be output by the pixel parser.
8	X	PPA_EXTRA_SF: Set when CSI-PPA receives an SF when it is expecting an EF. This happens when the EF of the frame gets corrupted before the arriving CSI. CSI-PPA will insert a fake EF and drop the current frame with Correct SF.
7	X	PPA_SHORT_FRAME: Set when CSI-PPA receives a short frame. This bit gets set even if CSI_PPA_PAD_FRAME specifies that short frames are to be padded to the correct line length.
6	X	PPA_STMERR: Stream Error. Set when the control output of PPA does not follow the correct sequence. The correct sequence for CSI is: SF -> (SL_DATA or EF), SL_DATA -> (DATA or EL_DATA), DATA -> EL_DATA, EL_DATA -> (SL_DATA or EF), EF -> SF. Stream Errors can be caused by receiving a corrupted stream.
5	X	PPA_FIFO_OVRF: FIFO Overflow. Set when the FIFO that is feeding packets to PPA overflows.
4	X	PPA_PL_CRC_ERR: PayLoad CRC Error. Set when a packet that was processed by PPA had a payload CRC error.
3	X	PPA_SL_PKT_DROPPED: Short Line Packet Dropped. Set when an incoming packet gets dropped because the input FIFO level reaches CSI_A_SKIP_PACKET_THRESHOLD when padding a short line.
2	X	PPA_SL_PROCESSED: Short Line Processed. Set when a line with a payload that is shorter than its packet header word count is processed by PPA.
1	X	PPA_ILL_WD_CNT: Illegal Word Count. Set when a line with a word count that does not generate an integer number of pixels (Unused bytes at the end of payload) is processed by PPA.
0	X	PPA_HDR_ERR_COR: Header Error Corrected. Set when a packet that was processed by PPA has a single bit header error. This error will be detected by the headers ECC, and corrected by it if header error correction is enabled (CSI_A_HEADER_EC_ENABLE = 0). This flag will be set and the packet will be processed even if the error is not corrected.

30.13.21 CSI_CSI_CIL_STATUS_0

CSI Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x2 to CSI_CIL_STATUS will clear only CILA_SOT_MB_ERR.

Offset: 0x21f | Byte Offset: 0x87c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
22	X	CILB_ESC_DATA_REC: Escape Mode Data Received. Set when CIL-B receives an Escape Mode Data Byte. The Data Byte can be read from bits 23:16 of ESCAPE_MODE_DATA. This status bit will also be cleared when CILB_ESC_CMD_REC is set.
21	X	CILB_ESC_CMD_REC: Escape Mode Command Received. Set when CIL-B receives an Escape Mode Command byte. The Command Byte can be read from bits 23:16 of ESCAPE_MODE_COMMAND.
20	X	CILB_CTRL_ERR: Control Error. Set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00)..
19	X	CILB_ESC_ENTRY_ERR: Escape Mode Entry Error. Set when CIL-B detects an Escape Mode Entry Error. The Escape mode command byte will not be received.
18	X	CILB_SYNC_ESC_ERR: Sync Escape Error. Set when CIL-B detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command or Data Byte.
17	X	CILB_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-B detects a multi-bit start of transmission byte error in one of the packet's SOT bytes. The packet will be discarded.
16	X	CILB_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-B detects a single bit error in one of the packet's start of transmission bytes. The packet will be sent to CSI-B for processing.
15	X	MIPI_AUTO_CAL_DONE: MIPI Auto Calibrate done. This bit field has been moved to armipi_cal. The definition is maintained here for backward compatibility.
6	X	CILA_ESC_DATA_REC: Escape Mode Data Received. Set when CIL-A receives an Escape Mode Data byte. The Data Byte can be read from bits 7:0 of ESCAPE_MODE_DATA. This status bit will also be cleared when CILA_ESC_CMD_REC is set.
5	X	CILA_ESC_CMD_REC: Escape Mode Command Received. Set when CIL-A receives an Escape Mode Command byte. The Command Byte can be read from bits 7:0 of ESCAPE_MODE_COMMAND.
4	X	CILA_CTRL_ERR: Control Error. Set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).
3	X	CILA_ESC_ENTRY_ERR: Escape Mode Entry Error. Set when CIL-A detects an escape mode entry error. The Escape mode command byte will not be received.
2	X	CILA_SYNC_ESC_ERR: Sync Escape Error. Set when CIL-A detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command or Data Byte.
1	X	CILA_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-A detects a multi-bit start of transmission byte error in one of the packet's SOT bytes. The packet will be discarded.
0	X	CILA_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-A detects a single bit error in one of the packet's Start of Transmission bytes. The packet will be sent to the CSI-A for processing.

30.13.22 CSI_CSI_PIXEL_PARSER_INTERRUPT_MASK_0

CSI Pixel Parser Interrupt Mask

Offset: 0x220 | Byte Offset: 0x880 | Read/Write: R/W | Reset: 0b00xx00000000000000xx000000000000

Bit	Reset	Description
31	0x0	HPH_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPH_UNC_HDR_ERR. Generate an interrupt when HPH_UNC_HDR_ERR is set. 0 = DISABLED: Do not generate an interrupt when HPH_UNC_HDR_ERR is set. 1 = ENABLED
30	0x0	HPV_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPV_UNC_HDR_ERR. Generate an interrupt when HPV_UNC_HDR_ERR is set. 0 = DISABLED: Do not generate an interrupt when HPV_UNC_HDR_ERR is set. 1 = ENABLED
26	0x0	PPB_SPARE_STATUS_1_INT_MASK: Interrupt Mask for PPB_SPARE_STATUS_1. Generate an interrupt when PPB_SPARE_STATUS_1 is set. 0 = DISABLED: Do not generate an interrupt when PPB_SPARE_STATUS_1 is set. 1 = ENABLED
25	0x0	PPB_INTERFRAME_LINE_INT_MASK: Interrupt Mask for PPB_INTERFRAME_LINE. Generate an interrupt when PPB_INTERFRAME_LINE is set. 0 = DISABLED: Do not generate an interrupt when PPB_INTERFRAME_LINE is set. 1 = ENABLED
24	0x0	PPB_EXTRA_SF_INT_MASK: Interrupt Mask for PPB_EXTRA_SF. Generate an interrupt when PPB_EXTRA_SF is set. 0 = DISABLED: Do not generate an interrupt when PPB_EXTRA_SF is set. 1 = ENABLED
23	0x0	PPB_SHORT_FRAME_INT_MASK: Interrupt Mask for PPB_SHORT_FRAME. Generate an interrupt when PPB_SHORT_FRAME is set. 0 = DISABLED: Do not generate an interrupt when PPB_SHORT_FRAME is set. 1 = ENABLED
22	0x0	PPB_STMERR_INT_MASK: Interrupt Mask for PPB_STMERR. Generate an interrupt when PPB_STMERR is set. 0 = DISABLED: Do not generate an interrupt when PPB_STMERR is set. 1 = ENABLED
21	0x0	PPB_FIFO_OVRF_INT_MASK: Interrupt Mask for PPB_FIFO_OVRF. Generate an interrupt when PPB_FIFO_OVRF is set. 0 = DISABLED: Do not generate an interrupt when PPB_FIFO_OVRF is set. 1 = ENABLED
20	0x0	PPB_PL_CRC_ERR_INT_MASK: Interrupt Mask for PPB_PL_CRC_ERR. Generate an interrupt when PPB_PL_CRC_ERR is set. 0 = DISABLED: Do not generate an interrupt when PPB_PL_CRC_ERR is set. 1 = ENABLED
19	0x0	PPB_SL_PKT_DROPPED_INT_MASK: Interrupt Mask for PPB_SL_PKT_DROPPED. Generate an interrupt when PPB_SL_PKT_DROPPED is set. 0 = DISABLED: Do not generate an interrupt when PPB_SL_PKT_DROPPED is set. 1 = ENABLED
18	0x0	PPB_SL_PROCESSED_INT_MASK: Interrupt Mask for PPB_SL_PROCESSED. Generate an interrupt when PPB_SL_PROCESSED is set. 0 = DISABLED: Do not generate an interrupt when PPB_SL_PROCESSED is set. 1 = ENABLED
17	0x0	PPB_ILL_WD_CNT_INT_MASK: Interrupt Mask for PPB_ILL_WD_CNT. Generate an interrupt when PPB_ILL_WD_CNT is set. 0 = DISABLED: Do not generate an interrupt when PPB_ILL_WD_CNT is set. 1 = ENABLED
16	0x0	PPB_HDR_ERR_COR_INT_MASK: Interrupt Mask for PPB_HDR_ERR_COR. Generate an interrupt when PPB_HDR_ERR_COR is set. 0 = DISABLED: Do not generate an interrupt when PPB_HDR_ERR_COR is set. 1 = ENABLED

Bit	Reset	Description
15	0x0	HPB_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPB_UNC_HDR_ERR. Generate an interrupt when HPB_UNC_HDR_ERR is set. 0 = DISABLED: Do not generate an interrupt when HPB_UNC_HDR_ERR is set. 1 = ENABLED
14	0x0	HPA_UNC_HDR_ERR_INT_MASK: Interrupt Mask for HPA_UNC_HDR_ERR. Generate an interrupt when HPA_UNC_HDR_ERR is set. 0 = DISABLED: Do not generate an interrupt when HPA_UNC_HDR_ERR is set. 1 = ENABLED
10	0x0	PPA_SPARE_STATUS_1_INT_MASK: Interrupt Mask for PPA_SPARE_STATUS_1. Generate an interrupt when PPA_SPARE_STATUS_1 is set. 0 = DISABLED: Do not generate an interrupt when PPA_SPARE_STATUS_1 is set. 1 = ENABLED
9	0x0	PPA_INTERFRAME_LINE_INT_MASK: Interrupt Mask for PPA_INTERFRAME_LINE. Generate an interrupt when PPA_INTERFRAME_LINE is set. 0 = DISABLED: Do not generate an interrupt when PPA_INTERFRAME_LINE is set. 1 = ENABLED
8	0x0	PPA_EXTRA_SF_INT_MASK: Interrupt Mask for PPA_EXTRA_SF. Generate an interrupt when PPA_EXTRA_SF is set. 0 = DISABLED: Do not generate an interrupt when PPA_EXTRA_SF is set. 1 = ENABLED
7	0x0	PPA_SHORT_FRAME_INT_MASK: Interrupt Mask for PPA_SHORT_FRAME. Generate an interrupt when PPA_SHORT_FRAME is set. 0 = DISABLED: Do not generate an interrupt when PPA_SHORT_FRAME is set. 1 = ENABLED
6	0x0	PPA_STMERR_INT_MASK: Interrupt Mask for PPA_STMERR. Generate an interrupt when PPA_STMERR is set. 0 = DISABLED: Do not generate an interrupt when PPA_STMERR is set. 1 = ENABLED
5	0x0	PPA_FIFO_OVRF_INT_MASK: Interrupt Mask for PPA_FIFO_OVRF. Generate an interrupt when PPA_FIFO_OVRF is set. 0 = DISABLED: Do not generate an interrupt when PPA_FIFO_OVRF is set. 1 = ENABLED
4	0x0	PPA_PL_CRC_ERR_INT_MASK: Interrupt Mask for PPA_PL_CRC_ERR. Generate an interrupt when PPA_PL_CRC_ERR is set. 0 = DISABLED: Do not generate an interrupt when PPA_PL_CRC_ERR is set. 1 = ENABLED
3	0x0	PPA_SL_PKT_DROPPED_INT_MASK: Interrupt Mask for PPA_SL_PKT_DROPPED. Generate an interrupt when PPA_SL_PKT_DROPPED is set. 0 = DISABLED: Do not generate an interrupt when PPA_SL_PKT_DROPPED is set. 1 = ENABLED
2	0x0	PPA_SL_PROCESSED_INT_MASK: Interrupt Mask for PPA_SL_PROCESSED. Generate an interrupt when PPA_SL_PROCESSED is set. 0 = DISABLED: Do not generate an interrupt when PPA_SL_PROCESSED is set. 1 = ENABLED
1	0x0	PPA_ILL_WD_CNT_INT_MASK: Interrupt Mask for PPA_ILL_WD_CNT. Generate an interrupt when PPA_ILL_WD_CNT is set. 0 = DISABLED: Do not generate an interrupt when PPA_ILL_WD_CNT is set. 1 = ENABLED
0	0x0	PPA_HDR_ERR_COR_INT_MASK: Interrupt Mask for PPA_HDR_ERR_COR. Generate an interrupt when PPA_HDR_ERR_COR is set. 0 = DISABLED: Do not generate an interrupt when PPA_HDR_ERR_COR is set. 1 = ENABLED

30.13.23 CSI_CSI_CIL_INTERRUPT_MASK_0

CSI Control and Interface Logic Interrupt Mask

Offset: 0x221 | Byte Offset: 0x884 | Read/Write: R/W | Reset: 0bxxxxxx0000000000xxxxxx0000000000

Bit	Reset	Description
22	0x0	CILB_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILB_ESC_DATA_REC. Generate an interrupt when CILB_ESC_DATA_REC is set. 0 = DISABLED: Do not generate an interrupt when CILB_ESC_DATA_REC is set. 1 = ENABLED
21	0x0	CILB_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILB_ESC_CMD_REC. Generate an interrupt when CILB_ESC_CMD_REC is set. 0 = DISABLED: Do not generate an interrupt when CILB_ESC_CMD_REC is set. 1 = ENABLED
20	0x0	CILB_CTRL_ERR_INT_MASK: Interrupt Mask for CILB_CTRL_ERR. Generate an interrupt when CILB_CTRL_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILB_CTRL_ERR is set. 1 = ENABLED
19	0x0	CILB_ESC_ENTRY_ERR_INT_MASK: Interrupt Mask for CILB_ESC_ENTRY_ERR. Generate an interrupt when CILB_ESC_ENTRY_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILB_ESC_ENTRY_ERR is set. 1 = ENABLED
18	0x0	CILB_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILB_SYNC_ESC_ERR. Generate an interrupt when CILB_SYNC_ESC_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILB_SYNC_ESC_ERR is set. 1 = ENABLED
17	0x0	CILB_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_MB_ERR. Generate an interrupt when CILB_SOT_MB_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILB_SOT_MB_ERR is set. 1 = ENABLED
16	0x0	CILB_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_SB_ERR. Generate an interrupt when CILB_SOT_SB_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILB_SOT_SB_ERR is set. 1 = ENABLED
9	0x0	CILA_CLK_LANE_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CLK_CTRL_ERR. Generate an interrupt when CILA_CLK_CTRL_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILA_CLK_CTRL_ERR is set. 1 = ENABLED
6	0x0	CILA_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILA_ESC_DATA_REC. Generate an interrupt when CILA_ESC_DATA_REC is set. 0 = DISABLED: Do not generate an interrupt when CILA_ESC_DATA_REC is set. 1 = ENABLED
5	0x0	CILA_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILA_ESC_CMD_REC. Generate an interrupt when CILA_ESC_CMD_REC is set. 0 = DISABLED: Do not generate an interrupt when CILA_ESC_CMD_REC is set. 1 = ENABLED
4	0x0	CILA_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CTRL_ERR. Generate an interrupt when CILA_CTRL_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILA_CTRL_ERR is set. 1 = ENABLED
3	0x0	CILA_ESC_ENTRY_ERR_INT_MASK: Interrupt Mask for CILA_ESC_ENTRY_ERR. Generate an interrupt when CILA_ESC_ENTRY_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILA_ESC_ENTRY_ERR is set. 1 = ENABLED
2	0x0	CILA_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILA_SYNC_ESC_ERR. Generate an interrupt when CILA_SYNC_ESC_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILA_SYNC_ESC_ERR is set. 1 = ENABLED

Bit	Reset	Description
1	0x0	CILA_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_MB_ERR. Generate an interrupt when CILA_SOT_MB_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILA_SOT_MB_ERR is set. 1 = ENABLED
0	0x0	CILA_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_SB_ERR. Generate an interrupt when CILA_SOT_SB_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILA_SOT_SB_ERR is set. 1 = ENABLED

30.13.24 CSI_CSI_READONLY_STATUS_0

CSI Read Only Status

This register is used to return CSI read only status.

Offset: 0x222 | Byte Offset: 0x888 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
1	X	CSI_PPB_ACTIVE: One only when Pixel Parser B is capturing frame data.
0	X	CSI_PPA_ACTIVE: One only when Pixel Parser A is capturing frame data.

30.13.25 CSI_ESCAPE_MODE_COMMAND_0

Escape Mode Command

This register is used to receive escape mode command bytes from CIL-A and CIL-B.

Offset: 0x223 | Byte Offset: 0x88c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	CILD_ESC_CMD_BYTE: CIL-D Escape Mode Command Byte. This is the 8-bit entry command that was received by CIL-D, during the last escape mode sequence. This command byte can only be assumed to be valid when the CILD_ESC_CMD_REC status bit is set.
23:16	X	CILB_ESC_CMD_BYTE: CIL-B Escape Mode Command Byte. This is the 8-bit entry command that was received by CIL-C, during the last escape mode sequence. This command byte can only be assumed to be valid when the CILC_ESC_CMD_REC status bit is set.
15:8	X	CILC_ESC_CMD_BYTE: CIL-C Escape Mode Command Byte. This is the 8-bit entry command that was received by CIL-B, during the last escape mode sequence. This command byte can only be assumed to be valid when the CILB_ESC_CMD_REC status bit is set.
7:0	X	CILA_ESC_CMD_BYTE: CIL-A Escape Mode Command Byte. This is the 8-bit entry command that was received by CIL-A, during the last escape mode sequence. CIL-A monitors Byte Lane 0 only for escape mode sequences. This command byte can only be assumed to be valid when the CILA_ESC_CMD_REC status bit is set.

30.13.26 CSI_ESCAPE_MODE_DATA_0

Escape Mode Data

This register is used to receive escape mode data bytes from CIL-A and CIL-B.

Offset: 0x224 | Byte Offset: 0x890 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	CILD_ESC_DATA_BYTE: CIL-D Escape Mode Data Byte. When read, this field returns the last Escape Mode Data byte that was received by CIL-B. Escape Mode Data bytes are the bytes that are received in Escape Mode after receiving the Escape Mode Command. These bytes can be used to implement MIPI's CSI Specs Low Power Data Transmission. This field is only valid when the status bit, CILD_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte come in.
23:16	X	CILB_ESC_DATA_BYTE: CIL-B Escape Mode Data Byte. When read, this field returns the last Escape Mode Data byte that was received by CIL-B. Escape Mode Data bytes are the bytes that are received in Escape Mode after receiving the Escape Mode Command. These bytes can be used to implement MIPI's CSI Specs Low Power Data Transmission. This field is only valid when the status bit, CILB_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte come in.
15:8	X	CILC_ESC_DATA_BYTE: CIL-C Escape Mode Data Byte. When read, this field returns the last Escape Mode Data byte that was received by CIL-B. Escape Mode Data bytes are the bytes that are received in Escape Mode after receiving the Escape Mode Command. These bytes can be used to implement MIPI's CSI Specs Low Power Data Transmission. This field is only valid when the status bit, CILB_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte come in.
7:0	X	CILA_ESC_DATA_BYTE: CIL-A Escape Mode Data Byte. When read, this field returns the last Escape Mode Data byte that was received by CIL-A. Escape Mode Data bytes are the bytes that are received in Escape Mode after receiving the Escape Mode Command. These bytes can be used to implement MIPI's CSI Specs Low Power Data Transmission. This field is only valid when the status bit, CILA_ESC_DATA_REC, is set, and will be overwritten by the next Escape Mode data byte if not read before the next byte come in.

30.13.27 CSI_CILA_PAD_CONFIG0_0

CIL-A Pad Configuration 0

Offset: 0x225 | Byte Offset: 0x894 | Read/Write: R/W | Reset: 0b0000000000000010x000x0000000x111

Bit	Reset	Description
31	0x0	PAD_CILA_E_TXBW: Increase bandwidth of output driver.
30:28	0x0	PAD_CILA_SLEWDNADJ: Pull-down slew rate adjust. Default = 000. From 000 -> 011, slew rate increases 100 is the same as 000 From 100->111, skew rate decreases.
27:25	0x0	PAD_CILA_SLEWUPADJ: Pull-up slew rate adjust. Default = 000. From 000 -> 011, slew rate increases 100 is the same as 000 From 100->111, skew rate decreases.
24:22	0x0	PAD_CILA_LPDNADJ: Driver pull-down impedance control. 00: 130 ohms, default 01: 110 ohms 10: 130 ohms, same as 00 11: 150 ohms

Bit	Reset	Description
21:19	0x0	PAD_CILA_LPUPADJ: Driver pull-up impedance control. 00: 130 ohms, default 01: 110 ohms 10: 130 ohms, same as 00 11: 150 ohms
18:17	0x1	PAD_AB_BK_MODE: 00: two independent 2x bricks 01: one 4x brick, received clock from partition A is used. Clock from partition B is not used 10: one 4x brick, received clock from partition B is used. Clock from partition A is not used 11: illegal
16	0x0	PAD_CILA_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILA_INADJ1: bit 1 input delay trimmer, each tap delays 20 ps
10:8	0x0	PAD_CILA_INADJ0: bit 0 input delay trimmer, each tap delays 20 ps
7	0x0	PAD_CILA_PDVCLAMP: Power-down regulator which supplies current to the serializer/deserializer logic
6:4	0x0	PAD_CILA_INADJCLK: Clock bit input delay trimmer, each tap delays 20 ps
2	0x1	PAD_CILA_PDIO_CLK: Power down for clock bit, including drivers, receivers, and contention detectors
1:0	0x3	PAD_CILA_PDIO: Power down for each data bit, including drivers, receivers, and contention detectors

30.13.28 CSI_CILA_PAD_CONFIG1_0

CIL-A Pad Configuration 4

Offset: 0x226 | Byte Offset: 0x898 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILA_SPARE_RO: Spare Read only bits for CILA Config
15:8	RW	0x0	PAD_CILA_SPARE: Spare bits for CILA Config. PAD_CILA_SPARE[15] is used to disable the CSI-A RTL code that blocks FIFO pushes that are past the end of the line packet. 0: Disabled 1: Push blocking enabled
7:6	RW	0x0	PAD_CILA_PEMPD_CLK: Enable data bit HS driver pull-down pre-emphasis. 00: no pre-emphasis 11: max
5:4	RW	0x0	PAD_CILA_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis 00: no pre-emphasis 11: max
3:2	RW	0x0	PAD_CILA_PEMPD: Enable data bit HS driver pull down pre-emphasis 00: no pre-emphasis 11: max
1:0	RW	0x0	PAD_CILA_PEMPU: Enable data bit HS driver pull up pre-emphasis 00: no pre-emphasis 11: max

30.13.29 CSI_CILB_PAD_CONFIG0_0

CIL-B Pad Configuration 0

Offset: 0x227 | Byte Offset: 0x89c | Read/Write: R/W | Reset: 0b0000000000000xx0x000x0000000x111

Bit	Reset	Description
31	0x0	PAD_CILB_E_TXBW: Increase bandwidth of output driver
30:28	0x0	PAD_CILB_SLEWDNADJ: Pull-down slew rate adjust. Default: 000. From 000 -> 011, slew rate increases 100 is the same as 000 From 100->111, skew rate decreases.
27:25	0x0	PAD_CILB_SLEWUPADJ: Pull-up slew rate adjust. Default: 000. From 000 -> 011, slew rate increases 100 is the same as 000 From 100->111, skew rate decreases.
24:22	0x0	PAD_CILB_LPDNADJ: Driver pull-down impedance control. 00: 130 ohms, default 01: 110 ohms 10: 130 ohms, same as 00 11: 150 ohms
21:19	0x0	PAD_CILB_LPUPADJ: Driver pull-up impedance control. 00: 130 ohms, default 01: 110 ohms 10: 130 ohms, same as 00 11: 150 ohms
16	0x0	PAD_CILB_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILB_INADJ1: Bit 1 input delay trimmer, each tap delays 20 ps
10:8	0x0	PAD_CILB_INADJ0: Bit 0 input delay trimmer, each tap delays 20 ps
7	0x0	PAD_CILB_PDVCLAMP: Power-down regulator which supplies current to serializer/deserializer logic
6:4	0x0	PAD_CILB_INADJCLK: Clock bit input delay trimmer, each tap delays 20 ps
2	0x1	PAD_CILB_PDIO_CLK: Power down for clock bit, including drivers, receivers, and contention detectors
1:0	0x3	PAD_CILB_PDIO: Power down for each data bit, including drivers, receivers, and contention detectors

30.13.30 CSI_CILB_PAD_CONFIG1_0

CIL-B Pad Configuration 4

Offset: 0x228 | Byte Offset: 0x8a0 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxx0000000000000000

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILB_SPARE_RO: Spare Read only bits for CILB Config
15:8	RW	0x0	PAD_CILB_SPARE: Spare bits for CILB Config. PAD_CILB_SPARE[15] are used to disable the CSI-B RTL code that blocks FIFO pushes that are past the end of the line packet. 0: disabled 1: push blocking enabled

Bit	R/W	Reset	Description
7:6	RW	0x0	PAD_CILB_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis. 00: no pre-emphasis 11: max
5:4	RW	0x0	PAD_CILB_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis. 00: no pre-emphasis 11: max
3:2	RW	0x0	PAD_CILB_PEMPD: Enable data bit HS driver pull down pre-emphasis. 00: no pre-emphasis 11: max
1:0	RW	0x0	PAD_CILB_PEMPU: Enable data bit HS driver pull up pre-emphasis. 00: no pre-emphasis 11: max

30.13.31 CSI_CIL_PAD_CONFIG0_0

CIL Pad Configuration 0

Offset: 0x229 | Byte Offset: 0x8a4 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx00000000000x000xx10

Bit	Reset	Description
18:16	0x0	PAD_CIL_VCLAMP_LEVEL: This field is deprecated VI register space and has been moved to the armipi_cal space.
15:8	0x0	PAD_CIL_SPARE: Spare bit for CIL BIAS Config PAD_CIL_SPARE[7] is used to flush VI's Y-FIFO when it is used as a stream source for one of the Pixel Parsers. Setting PAD_CIL_SPARE[7] to 1 will hold vi2csi_host_stall low, which forces VI's Y-FIFO to be purged. PAD_CIL_SPARE[7] must be low for the pixel parser to receive source data from VI's Y-FIFO.
6:4	0x0	PAD_CIL_VADJ: This field is deprecated VI register space and has been moved to the armipi_cal space.
1	0x1	PAD_CIL_PDVREG: This field is deprecated VI register space and has been moved to the armipi_cal space.
0	0x0	PAD_CIL_VBYPASS: This field is deprecated VI register space and has been moved to the armipi_cal space.

30.13.32 CSI_MIPIBIAS_PAD_CONFIG0_0

All fields of this register are deprecated. Writes are ignored; Reads return zero.

MIPI Bias Pad Configuration Register

Offset: 0x235 | Byte Offset: 0x8d4 | Read/Write: R/W | Reset: 0bxxxxx000xxxxx000xxxxx000xxxxx000

Bit	Reset	Description
26:24	0x0	PAD_TEST_SEL
18:16	0x0	PAD_DRIV_DN_REF
10:8	0x0	PAD_DRIV_UP_REF
2:0	0x0	PAD_TERM_REF

30.13.33 CSI_CSI_CILA_STATUS_0

CSI-CILA Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x20 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x236 | Byte Offset: 0x8d8 | Read/Write: R/O | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18	X	CILA_DATA_LANE1_CTRL_ERR: Control Error, set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning.
17	X	CILA_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-A detects a multi-bit start of transmission byte error in one of the packet's SOT bytes on data lane-1. The packet will be discarded.
16	X	CILA_DATA_LANE0_CTRL_ERR: Control Error, set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning.
6	X	CILA_DATA_LANE0_CTRL_ERR: Control Error, set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning.
5	X	CILA_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-A detects a multi-bit start of transmission byte error in one of the packet's SOT bytes on data lane-0. The packet will be discarded.
4	X	CILA_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-A detects a single bit error in one of the packet's Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-A for processing.
0	X	CILA_CLK_LANE_CTRL_ERR: Control Error, set when CIL-A detects incorrect line state sequence on the clock lane.

30.13.34 CSI_CSI_CILB_STATUS_0

CSI-CILB Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x20 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x237 | Byte Offset: 0x8dc | Read/Write: R/O | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18	X	CILB_DATA_LANE1_CTRL_ERR: Control Error. Set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
17	X	CILB_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-B detects a multi-bit start of transmission byte error in one of the packet's SOT bytes on data lane 1. The packet will be discarded.
16	X	CILB_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-B detects a single bit error in one of the packet's Start of Transmission bytes on data lane 1. The packet will be sent to the CSI-B for processing.
6	X	CILB_DATA_LANE0_CTRL_ERR: Control Error, set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
5	X	CILB_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-B detects a multi-bit start of transmission byte error in one of the packet's SOT bytes on data lane 0. The packet will be discarded.
4	X	CILB_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-B detects a single bit error in one of the packet's Start of Transmission bytes on data lane 0. The packet will be sent to the CSI-B for processing.

Bit	Reset	Description
0	X	CILB_CLK_LANE_CTRL_ERR: Control Error, set when CIL-B detects an incorrect line state sequence on the clock lane

30.13.35 CSI_CSI_CILC_STATUS_0

CSI-CILC Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x20 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x239 | Byte Offset: 0x8e4 | Read/Write: R/O | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18	X	CILC_DATA_LANE1_CTRL_ERR: Control Error. Set when CIL-C detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning
17	X	CILC_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-C detects a multi-bit start of transmission byte error in one of the packet's SOT bytes on data lane-1. The packet will be discarded.
16	X	CILC_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-C detects a single bit error in one of the packet's Start of Transmission bytes on data lane-1. The packet will be sent to the CSI-C for processing.
6	X	CILC_DATA_LANE0_CTRL_ERR: Control Error. Set when CIL-C detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning.
5	X	CILC_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-C detects a multi bit start of transmission byte error in one of the packet's SOT bytes on data lane-0. The packet will be discarded.
4	X	CILC_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-C detects a single bit error in one of the packet's Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-C for processing.
0	X	CILC_CLK_LANE_CTRL_ERR: Control Error. Set when CIL-C detects incorrect line state sequence on the clock lane.

30.13.36 CSI_CSI_CILD_STATUS_0

CSI-CILD Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x20 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 0x23a | Byte Offset: 0x8e8 | Read/Write: R/O | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18	X	CILD_DATA_LANE1_CTRL_ERR: Control Error. Set when CIL-D detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning.
17	X	CILD_DATA_LANE1_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-D detects a multi-bit start of transmission byte error in one of the packet's SOT bytes on data lane-1. The packet will be discarded.
16	X	CILD_DATA_LANE1_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-D detects a single bit error in one of the packet's Start of Transmission bytes on data lane-1. The packet will be sent to the CSI-D for processing.
6	X	CILD_DATA_LANE0_CTRL_ERR: Control Error. Set when CIL-D detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning.

Bit	Reset	Description
5	X	CILD_DATA_LANE0_SOT_MB_ERR: Start of Transmission Multi Bit Error. Set when CIL-D detects a multi-bit start of transmission byte error in one of the packet's SOT bytes on data lane-0. The packet will be discarded.
4	X	CILD_DATA_LANE0_SOT_SB_ERR: Start of Transmission Single Bit Error. Set when CIL-D detects a single bit error in one of the packet's Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-D for processing.
0	X	CILD_CLK_LANE_CTRL_ERR: Control Error. Set when CIL-D detects incorrect line state sequence on the clock lane.

30.13.37 CSI_PHY_CILC_CONTROL0_0

CSI-C PHY and CIL Control

Offset: 0x23e | Byte Offset: 0x8f8 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxx0000xx0x0010

Bit	Reset	Description
11:8	0x0	CILC_CLK_SETTLE: Settle time for the clock lane when moving from LP to HS (LP11->LP01->LP00). This setting determines how many CSICIL clock cycles (102 MHz LP clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 CSICIL cycles for default value. So the programming value (0 to 15) of this field works as follows: 0 = 15 cilclk cycles (default internal delay) 1 = 15 - 7 (8 cilclk cycles) 2 = 15 - 6 (9 cilclk cycles) … 7 = 15 - 1 (14 cilclk cycles) 8 = 15 - 0 (default internal delay) 9 = 15 + 1 (16 cilclk cycles) … 15 = 15 + 7 (22 cilclk cycles)
5	0x0	CILC_BYPASS_LP_SEQ: The LP signals should sequence through LP11->LP01->LP00 states to indicate to CLOCK CIL about the mode switching to HS Rx mode. In case the camera is enabled earlier than CIL, it is highly likely that the camera sends this control sequence sooner than CIL can detect it. Enabling this bit allows the CLOCK CIL to overlook the LP control sequence and step in HS Rx mode directly looking at LP00 only.
3:0	0x2	CILC_THS_SETTLE: Settle time for data lane when moving from LP to HS (LP11->LP01->LP00). This setting determines how many csicil clock cycles (102 MHz LP clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

30.13.38 CSI_PHY_CILD_CONTROL0_0

CSI-D PHY and CIL Control

Offset: 0x23f | Byte Offset: 0x8fc | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxx0000xx0x0010

Bit	Reset	Description
11:8	0x0	CILD_CLK_SETTLE: Settle time for clock lane when moving from LP to HS (LP11->LP01->LP00). This setting determines how many CSICIL clock cycles (102 MHz LP clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 CSICIL cycles for default value. So the programming value (0 to 15) of this field works as follows: 0 = 15 cilclk cycles (default internal delay) 1 = 15 - 7 (8 cilclk cycles) 2 = 15 - 6 (9 cilclk cycles) … 7 = 15 - 1 (14 cilclk cycles)

Bit	Reset	Description
		8 = 15 - 0 (default internal delay) 9 = 15 + 1 (16 cilclk cycles) … 15 = 15 + 7 (22 cilclk cycles)
5	0x0	CILD_BYPASS_LP_SEQ: See CILC_BYPASS_LP_SEQ above.
3:0	0x2	CILD_THS_SETTLE: Settle time for data lane when moving from LP to HS (LP11->LP01->LP00). This setting determines how many CSICIL clock cycles (102 MHz LP clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

30.13.39 CSI_PHY_CILE_CONTROL0_0

CSI-E PHY and CIL Control

Offset: 0x240 | Byte Offset: 0x90c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxx0000xx0x0010

Bit	Reset	Description
11:8	0x0	CILE_CLK_SETTLE: Settle time for the clock lane when moving from LP to HS (LP11->LP01->LP00). This setting determines how many CSICIL clock cycles (102 MHz LP clock cycles) to wait after LP00. Hardware uses an internal delay of ~15 CSICIL cycles for default value. So the programming value (0 to 15) of this field works as follows: 0 = 15 cilclk cycles (default internal delay) 1 = 15 - 7 (8 cilclk cycles) 2 = 15 - 6 (9 cilclk cycles) … 7 = 15 - 1 (14 cilclk cycles) 8 = 15 - 0 (default internal delay) 9 = 15 + 1 (16 cilclk cycles) … 15 = 15 + 7 (22 cilclk cycles)
5	0x0	CILE_BYPASS_LP_SEQ: The LP signals should sequence through LP11->LP01->LP00 states to indicate to CLOCK CIL about the mode switching to HS Rx mode. In case the camera is enabled earlier than CIL, it is highly likely that the camera sends this control sequence sooner than CIL can detect it. Enabling this bit allows the CLOCK CIL to overlook the LP control sequence and step in HS Rx mode directly looking at LP00 only.
3:0	0x2	CILE_THS_SETTLE: Settle time for data lane when moving from LP to HS (LP11->LP01->LP00). This setting determines how many csicil clock cycles (102 MHz LP clock cycles) to wait, after LP00, before starting to look at the data. 0xF is an illegal value for this field 85ns + 6 * UI < (Ths-settle-programmed + 5) * csicil_clk_period < 145ns + 10 * UI

30.13.40 CSI_CSI_CIL_STATUS_2_0

CSI Control and Interface Logic Status

Each status bit is cleared to zero when its bit position is written with one. For example, writing 0x2 to CSI_CIL_STATUS will clear only CILA_SOT_MB_ERR.

Offset: 0x244 | Byte Offset: 0x910 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
22	X	CILD_ESC_DATA_REC: Escape Mode Data Received, set when CIL-B receives an Escape Mode Data byte. The Data Byte can be read from bits 23:16 of ESCAPE_MODE_DATA. This status bit will also be cleared when CILB_ESC_CMD_REC is set.

Bit	Reset	Description
21	X	CILD_ESC_CMD_REC: Escape Mode Command Received, set when CIL-B receives an Escape Mode Command byte. The Command Byte can be read from bits 23:16 of ESCAPE_MODE_COMMAND.
20	X	CILD_CTRL_ERR: Control Error, set when CIL-B detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).
19	X	CILD_ESC_ENTRY_ERR: Escape Mode Entry Error, set when CIL-B detects an Escape Mode Entry Error. The Escape mode command byte will not be received.
18	X	CILD_SYNC_ESC_ERR: Sync Escape Error, set when CIL-B detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command or Data Byte.
17	X	CILD_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-B detects a multi-bit start of transmission byte error in one of the packet's SOT bytes. The packet will be discarded.
16	X	CILD_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-B detects a single bit error in one of the packet's start of transmission bytes. The packet will be sent to CSI-B for processing.
6	X	CILC_ESC_DATA_REC: Escape Mode Data Received, set when CIL-A receives an Escape Mode Data byte. The Data Byte can be read from bits 7:0 of ESCAPE_MODE_DATA. This status bit will also be cleared when CILA_ESC_CMD_REC is set.
5	X	CILC_ESC_CMD_REC: Escape Mode Command Received, set when CIL-A receives an Escape Mode Command byte. The Command Byte can be read from bits 7:0 of ESCAPE_MODE_COMMAND.
4	X	CILC_CTRL_ERR: Control Error, set when CIL-A detects LP state 01 or 10 followed by a stop state (LP11) instead of transitioning into the Escape mode or Turn Around mode (LP00).
3	X	CILC_ESC_ENTRY_ERR: Escape Mode Entry Error, set when CIL-A detects an escape mode entry error. The Escape mode command byte will not be received.
2	X	CILC_SYNC_ESC_ERR: Sync Escape Error, set when CIL-A detects that the wrong (non-multiple of 8) number of bits have been received for an Escape Command, or Data Byte.
1	X	CILC_SOT_MB_ERR: Start of Transmission Multi Bit Error, set when CIL-A detects a multi-bit start of transmission byte error in one of the packet's SOT bytes. The packet will be discarded.
0	X	CILC_SOT_SB_ERR: Start of Transmission Single Bit Error, set when CIL-A detects a single bit error in one of the packet's Start of Transmission bytes. The packet will be sent to the CSI-A for processing.

30.13.41 CSI_CSI_CIL_INTERRUPT_MASK_2_0

CSI Control and Interface Logic Interrupt Mask

Offset: 0x245 | Byte Offset: 0x914 | Read/Write: R/W | Reset: 0bxxxxxx0xx0000000xxxxxx0xx0000000

Bit	Reset	Description
25	0x0	CILD_CLK_LANE_CTRL_ERR_INT_MASK: Interrupt Mask for CILB_CLK_CTRL_ERR. Generate an interrupt when CILB_CLK_CTRL_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILB_CLK_CTRL_ERR is set. 1 = ENABLED
22	0x0	CILD_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILB_ESC_DATA_REC. Generate an interrupt when CILB_ESC_DATA_REC is set. 0 = DISABLED: Do not generate an interrupt when CILB_ESC_DATA_REC is set. 1 = ENABLED
21	0x0	CILD_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILB_ESC_CMD_REC. Generate an interrupt when CILB_ESC_CMD_REC is set.

Bit	Reset	Description
		0 = DISABLED: Do not generate an interrupt when CILB_ESC_CMD_REC is set. 1 = ENABLED
20	0x0	CILD_CTRL_ERR_INT_MASK: Interrupt Mask for CILB_CTRL_ERR. Generate an interrupt when CILB_CTRL_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILB_CTRL_ERR is set. 1 = ENABLED
19	0x0	CILD_ESC_ENTRY_ERR_INT_MASK: Interrupt Mask for CILB_ESC_ENTRY_ERR. Generate an interrupt when CILB_ESC_ENTRY_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILB_ESC_ENTRY_ERR is set. 1 = ENABLED
18	0x0	CILD_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILB_SYNC_ESC_ERR. Generate an interrupt when CILB_SYNC_ESC_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILB_SYNC_ESC_ERR is set. 1 = ENABLED
17	0x0	CILD_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_MB_ERR. Generate an interrupt when CILB_SOT_MB_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILB_SOT_MB_ERR is set. 1 = ENABLED
16	0x0	CILD_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILB_SOT_SB_ERR. Generate an interrupt when CILB_SOT_SB_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILB_SOT_SB_ERR is set. 1 = ENABLED
9	0x0	CILC_CLK_LANE_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CLK_CTRL_ERR. Generate an interrupt when CILA_CLK_CTRL_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILA_CLK_CTRL_ERR is set. 1 = ENABLED
6	0x0	CILC_ESC_DATA_REC_INT_MASK: Interrupt Mask for CILA_ESC_DATA_REC. Generate an interrupt when CILA_ESC_DATA_REC is set. 0 = DISABLED: Do not generate an interrupt when CILA_ESC_DATA_REC is set. 1 = ENABLED
5	0x0	CILC_ESC_CMD_REC_INT_MASK: Interrupt Mask for CILA_ESC_CMD_REC. Generate an interrupt when CILA_ESC_CMD_REC is set. 0 = DISABLED: Do not generate an interrupt when CILA_ESC_CMD_REC is set. 1 = ENABLED
4	0x0	CILC_CTRL_ERR_INT_MASK: Interrupt Mask for CILA_CTRL_ERR. Generate an interrupt when CILA_CTRL_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILA_CTRL_ERR is set. 1 = ENABLED
3	0x0	CILC_ESC_ENTRY_ERR_INT_MASK: Interrupt Mask for CILA_ESC_ENTRY_ERR. Generate an interrupt when CILA_ESC_ENTRY_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILA_ESC_ENTRY_ERR is set. 1 = ENABLED
2	0x0	CILC_SYNC_ESC_ERR_INT_MASK: Interrupt Mask for CILA_SYNC_ESC_ERR. Generate an interrupt when CILA_SYNC_ESC_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILA_SYNC_ESC_ERR is set. 1 = ENABLED
1	0x0	CILC_SOT_MB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_MB_ERR. Generate an interrupt when CILA_SOT_MB_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILA_SOT_MB_ERR is set. 1 = ENABLED
0	0x0	CILC_SOT_SB_ERR_INT_MASK: Interrupt Mask for CILA_SOT_SB_ERR. Generate an interrupt when CILA_SOT_SB_ERR is set. 0 = DISABLED: Do not generate an interrupt when CILA_SOT_SB_ERR is set. 1 = ENABLED

30.13.42 CSI_CILC_PAD_CONFIG0_0

CIL-C Pad Configuration 0

Offset: 0x24a | Byte Offset: 0x928 | Read/Write: R/W | Reset: 0b0000000000000010x000x0000000x111

Bit	Reset	Description
31	0x0	PAD_CILC_E_TXBW: Increase bandwidth of output driver.
30:28	0x0	PAD_CILC_SLEWDNADJ: Pull-down slew rate adjust. Default = 000. From 000 -> 011, slew rate increases 100 is the same as 000. From 100->111, skew rate decreases.
27:25	0x0	PAD_CILC_SLEWUPADJ: Pull-up slew rate adjust. Default = 000. From 000 -> 011, slew rate increases 100 is the same as 000. From 100->111, skew rate decreases.
24:22	0x0	PAD_CILC_LPDNADJ: Driver pull-down impedance control. 00: 130 ohms, default 01: 110 ohms 10: 130 ohms, same as 00 11: 150 ohms
21:19	0x0	PAD_CILC_LPUPADJ: Driver pull up impedance control. 00: 130 ohms, default 01: 110 ohms 10: 130 ohms, same as 00 11: 150 ohms
18:17	0x1	PAD_CD_BK_MODE: 00: two independent 2x bricks 01: one 4x brick, received clock from partition C is used. Clock from partition D is not used 10: one 4x brick, received clock from partition D is used. Clock from partition C is not used 11: illegal
16	0x0	PAD_CILC_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILC_INADJ1: bit 1 input delay trimmer, each tap delays 20 ps
10:8	0x0	PAD_CILC_INADJ0: bit 0 input delay trimmer, each tap delays 20 ps
7	0x0	PAD_CILC_PDVCLAMP: Power-down regulator which supplies current to the serializer/deserializer logic
6:4	0x0	PAD_CILC_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
2	0x1	PAD_CILC_PDIO_CLK: Power down for clock bit, including drivers, receivers, and contention detectors
1:0	0x3	PAD_CILC_PDIO: Power down for each data bit, including drivers, receivers, and contention detectors

30.13.43 CSI_CILC_PAD_CONFIG1_0

CIL-C Pad Configuration 1

Offset: 0x24b | Byte Offset: 0x92c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
7:6	0x0	PAD_CILC_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis 00: no pre-emphasis 11: max

Bit	Reset	Description
5:4	0x0	PAD_CILC_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis 00: no pre-emphasis 11: max
3:2	0x0	PAD_CILC_PEMPD: Enable data bit HS driver pull down pre-emphasis 00: no pre-emphasis 11: max
1:0	0x0	PAD_CILC_PEMPU: Enable data bit HS driver pull up pre-emphasis 00: no pre-emphasis 11: max

30.13.44 CSI_CILD_PAD_CONFIG0_0

CIL-D Pad Configuration 0

Offset: 0x24c | Byte Offset: 0x930 | Read/Write: R/W | Reset: 0b0000000000000xx0x000x0000000x111

Bit	Reset	Description
31	0x0	PAD_CILD_E_TXBW: Increase bandwidth of output driver
30:28	0x0	PAD_CILD_SLEWDNADJ: Pull-down slew rate adjust. Default = 000. From 000 -> 011, slew rate increases 100 is the same as 000 From 100->111, skew rate decreases.
27:25	0x0	PAD_CILD_SLEWUPADJ: Pull-up slew rate adjust. Default = 000. From 000 -> 011, slew rate increases 100 is the same as 000 From 100->111, skew rate decreases.
24:22	0x0	PAD_CILD_LPDNADJ: Driver pull-down impedance control 00: 130 ohms; default 01: 110 ohms 10: 130 ohms, same as 00 11: 150 ohms
21:19	0x0	PAD_CILD_LPUPADJ: Driver pull-up impedance control. 00: 130 ohms, default 01: 110 ohms 10: 130 ohms, same as 00 11: 150 ohms
16	0x0	PAD_CILD_BANDWD_IN: Increase bandwidth of differential receiver
14:12	0x0	PAD_CILD_INADJ1: bit 1 input delay trimmer, each tap delays 20 ps
10:8	0x0	PAD_CILD_INADJ0: bit 0 input delay trimmer, each tap delays 20 ps
7	0x0	PAD_CILD_PDVCLAMP: Power-down regulator which supplies current to serializer/deserializer logic
6:4	0x0	PAD_CILD_INADJCLK: Clock bit input delay trimmer, each tap delays 20 ps
2	0x1	PAD_CILD_PDIO_CLK: Power down for clock bit, including drivers, receivers, and contention detectors
1:0	0x3	PAD_CILD_PDIO: Power down for each data bit, including drivers, receivers, and contention detectors

30.13.45 CSI_CILD_PAD_CONFIG1_0

CIL-D Pad Configuration 1

Offset: 0x24d | Byte Offset: 0x934 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
7:6	0x0	PAD_CILD_PEMPD_CLK: Enable data bit HS driver pull down pre-emphasis. 00: no pre-emphasis 11: max
5:4	0x0	PAD_CILD_PEMPU_CLK: Enable data bit HS driver pull up pre-emphasis. 00: no pre-emphasis 11: max
3:2	0x0	PAD_CILD_PEMPD: Enable data bit HS driver pull down pre-emphasis. 00: no pre-emphasis 11: max
1:0	0x0	PAD_CILD_PEMPU: Enable data bit HS driver pull up pre-emphasis. 00: no pre-emphasis 11: max

30.13.46 CSI_CILE_PAD_CONFIG0_0

CIL-E Pad Configuration 0

Offset: 0x24e | Byte Offset: 0x938 | Read/Write: R/W | Reset: 0b0000000000000xxx0xxxx0000000x1x1

Bit	Reset	Description
31	0x0	PAD_CILE_E_TXBW: Increase bandwidth of output driver
30:28	0x0	PAD_CILE_SLEWDNADJ: Pull down slew rate adjust. Default = 000. From 000 -> 011, slew rate increases 100 is the same as 000 From 100->111, skew rate decreases.
27:25	0x0	PAD_CILE_SLEWUPADJ: Pull up slew rate adjust. Default = 000. From 000 -> 011, slew rate increases 100 is the same as 000 From 100->111, skew rate decreases.
24:22	0x0	PAD_CILE_LPDNADJ: Driver pull-down impedance control. 00: 130 ohms, default 01: 110 ohms 10: 130 ohms, same as 00 11: 150 ohms
21:19	0x0	PAD_CILE_LPUPADJ: Driver pull-up impedance control. 00: 130 ohms, default 01: 110 ohms 10: 130 ohms, same as 00 11: 150 ohms
15	0x0	PAD_CILE_BANDWD_IN: Increase bandwidth of differential receiver
10:8	0x0	PAD_CILE_INADJ0: bit 0 input delay trimmer, each tap delays 20 ps
7	0x0	PAD_CILE_PDVCLAMP: Power down regulator which supplies current to serializer/deserializer logic
6:4	0x0	PAD_CILE_INADJCLK: Clock bit input delay trimmer, each tap delays 20 ps

Bit	Reset	Description
2	0x1	PAD_CILE_PDIO_CLK: Power down for clock bit, including drivers, receivers, and contention detectors
0	0x1	PAD_CILE_PDIO: Power down for each data bit, including drivers, receivers, and contention detectors

30.13.47 CSI_CILE_PAD_CONFIG1_0

CIL-E Pad Configuration 4

Offset: 0x24f | Byte Offset: 0x93c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
7:6	0x0	PAD_CILE_PEMPD_CLK: Enable data bit HS driver pull-down pre-emphasis. 00: no pre-emphasis 11: max
5:4	0x0	PAD_CILE_PEMPU_CLK: Enable data bit HS driver pull-up pre-emphasis. 00: no pre-emphasis 11: max
3:2	0x0	PAD_CILE_PEMPD: Enable data bit HS driver pull-down pre-emphasis. 00: no pre-emphasis 11: max
1:0	0x0	PAD_CILE_PEMPU: Enable data bit HS driver pull-up pre-emphasis. 00: no pre-emphasis 11: max

30.13.48 CSI_PATTERN_GENERATOR_CTRL_A_0

Offset: 0x250 | Byte Offset: 0x940 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000

Bit	Reset	Description
3:2	0x0	PG_MODE_A: Mode for Sensor A 0: DIRECT 1: SINE_LUT 2: PATCH 3: RSVD
1	0x0	PG_AUTO_INC_A: Automatic phase increment mode for sensor A
0	0x0	PG_ENABLE_A: Enable Pattern Generator for sensor A

30.13.49 CSI_PG_BLANK_A_0

Offset: 0x251 | Byte Offset: 0x944 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000100000001000

Bit	Reset	Description
15:8	0x8	PG_VBLANK_A: Vertical Blanking for Pattern Generator
7:0	0x8	PG_HBLANK_A: Horizontal Blanking for Pattern Generator

30.13.50 CSI_PG_PHASE_A_0

Offset: 0x252 | Byte Offset: 0x948 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxx00000000000000

Bit	Reset	Description
13:0	0x0	PG_PHASE_A: Initial Phase

30.13.51 CSI_PG_RED_FREQ_A_0

Offset: 0x253 | Byte Offset: 0x94c | Read/Write: R/W | Reset: 0bxx00000000000000xx00000000000000

Bit	Reset	Description
29:16	0x0	PG_RED_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_RED_HOR_INIT_FREQ_A: Initial horizontal frequency

30.13.52 CSI_PG_RED_FREQ_RATE_A_0

Offset: 0x254 | Byte Offset: 0x950 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15:8	0x0	PG_RED_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_RED_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

30.13.53 CSI_PG_GREEN_FREQ_A_0

Offset: 0x255 | Byte Offset: 0x954 | Read/Write: R/W | Reset: 0bxx00000000000000xx00000000000000

Bit	Reset	Description
29:16	0x0	PG_GREEN_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_GREEN_HOR_INIT_FREQ_A: Initial horizontal frequency

30.13.54 CSI_PG_GREEN_FREQ_RATE_A_0

Offset: 0x256 | Byte Offset: 0x958 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15:8	0x0	PG_GREEN_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_GREEN_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

30.13.55 CSI_PG_BLUE_FREQ_A_0

Offset: 0x257 | Byte Offset: 0x95c | Read/Write: R/W | Reset: 0bxx00000000000000xx00000000000000

Bit	Reset	Description
29:16	0x0	PG_BLUE_VERT_INIT_FREQ_A: Initial vertical frequency
13:0	0x0	PG_BLUE_HOR_INIT_FREQ_A: Initial horizontal frequency

30.13.56 CSI_PG_BLUE_FREQ_RATE_A_0

Offset: 0x258 | Byte Offset: 0x960 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15:8	0x0	PG_BLUE_VERT_FREQ_RATE_A: Rate of change of vertical frequency
7:0	0x0	PG_BLUE_HOR_FREQ_RATE_A: Rate of change of horizontal frequency

30.13.57 CSI_PATTERN_GENERATOR_CTRL_B_0

Offset: 0x25d | Byte Offset: 0x974 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000

Bit	Reset	Description
3:2	0x0	PG_MODE_B: Mode for Sensor B 0: DIRECT 1: SINE_LUT 2: PATCH 3: RSVD
1	0x0	PG_AUTO_INC_B: Automatic phase increment mode for sensor B
0	0x0	PG_ENABLE_B: Enable Pattern Generator for sensor B

30.13.58 CSI_PG_BLANK_B_0

Offset: 0x25e | Byte Offset: 0x978 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000100000001000

Bit	Reset	Description
15:8	0x8	PG_VBLANK_B: Vertical Blanking for Pattern Generator
7:0	0x8	PG_HBLANK_B: Horizontal Blanking for Pattern Generator

30.13.59 CSI_PG_PHASE_B_0

Offset: 0x25f | Byte Offset: 0x97c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxx00000000000000

Bit	Reset	Description
13:0	0x0	PG_PHASE_B: Initial Phase

30.13.60 CSI_PG_RED_FREQ_B_0

Offset: 0x260 | Byte Offset: 0x980 | Read/Write: R/W | Reset: 0bxx00000000000000xx00000000000000

Bit	Reset	Description
29:16	0x0	PG_RED_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_RED_HOR_INIT_FREQ_B: Initial horizontal frequency

30.13.61 CSI_PG_RED_FREQ_RATE_B_0

Offset: 0x261 | Byte Offset: 0x984 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15:8	0x0	PG_RED_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_RED_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

30.13.62 CSI_PG_GREEN_FREQ_B_0

Offset: 0x262 | Byte Offset: 0x988 | Read/Write: R/W | Reset: 0bxx00000000000000xx00000000000000

Bit	Reset	Description
29:16	0x0	PG_GREEN_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_GREEN_HOR_INIT_FREQ_B: Initial horizontal frequency

30.13.63 CSI_PG_GREEN_FREQ_RATE_B_0

Offset: 0x263 | Byte Offset: 0x98c | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15:8	0x0	PG_GREEN_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_GREEN_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

30.13.64 CSI_PG_BLUE_FREQ_B_0

Offset: 0x264 | Byte Offset: 0x990 | Read/Write: R/W | Reset: 0bxx00000000000000xx00000000000000

Bit	Reset	Description
29:16	0x0	PG_BLUE_VERT_INIT_FREQ_B: Initial vertical frequency
13:0	0x0	PG_BLUE_HOR_INIT_FREQ_B: Initial horizontal frequency

30.13.65 CSI_PG_BLUE_FREQ_RATE_B_0

Offset: 0x265 | Byte Offset: 0x994 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
15:8	0x0	PG_BLUE_VERT_FREQ_RATE_B: Rate of change of vertical frequency
7:0	0x0	PG_BLUE_HOR_FREQ_RATE_B: Rate of change of horizontal frequency

31.0 MIPI D-PHY CALIBRATION FOR CSI AND DSI

This section describes the MIPI D-PHY Calibration registers. This information is provided to aid in system debug and diagnostics. It is expected that NVIDIA® supplied drivers will normally be used to program these.

MIPI D-PHY blocks are used for both the DSI display interface and the CSI camera interface. Both interfaces are controlled by these registers.

31.1 MIPI-CAL Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

31.1.1 MIPI_CAL_MIPI_CAL_CTRL_0

Calibration Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x2a000000 (0bxx101010xxxxxxxxxxxxxxxxxxxx0xx00)

Bit	R/W	Reset	Description
29:26	RW	0xa	MIPI_CAL_NOISE_FLT: The DRIVRY and TERMRY signals coming from MIPI pads are utilized by the Calibration state machine for pad calibration. The drivry/termry comes from a noisy analog source, which might have some glitches. The filter in calibsm is sensitive to these noises. If the calibration done status does not show up, the sensitivity of the filter can be changed through these bits. Ideally, this has to be programmed in a range from 10 to 15. When MIPI_CAL_PRESCALE = 2'b00, this needs to be programmed between 2 to 5.
25:24	RW	0x2	MIPI_CAL_PRESCALE: Auto-Cal calibration step prescale: 00: when calibration step is 0.1 µs 01: when calibration step is 0.5 µs 10: when calibration step is 1.0 µs (default) 11: when calibration step is 1.5 µs This keeps the MIPI bias calibration step between 0.1-1.5 µs.
4	RW	CLK_GATED	MIPI_CAL_CLKEN_OVR: 0 = CLK_GATED 1 = CLK_ALWAYS_ON
1	RW	0x0	MIPI_CAL_AUTOCAL_EN: When set, calibration is triggered periodically. The Period is set using MIPI_CAL_AUTOCAL_CTRL0 register
0	RO	0x0	MIPI_CAL_STARTCAL: Writing a one to this bit starts the Calibration State machine. This bit must be set even if both overrides are set in order to latch in the override value.

31.1.2 MIPI_CAL_MIPI_CAL_AUTOCAL_CTRL0_0

Offset: 0x4 | Read/Write: R/W | Reset: 0xffffffff (0b11111111111111111111111111111111)

Bit	Reset	Description
31:0	-1	MIPI_CAL_AUTOCAL_PERIOD: Auto-calibration period in number of APB sys clk cycles.

31.1.3 MIPI_CAL_CIL_MIPI_CAL_STATUS_0

CIL MIPI Calibrate Status

Offset: 0x8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	MIPI_AUTO_CAL_DONE_DSID: Debug bit. MIPI Auto Calibrate done for DSID. Set when the auto calibrate sequence for DSI pad bricks is done. Write 1-to-clear.
30	X	MIPI_AUTO_CAL_DONE_DSIC: Debug bit. MIPI Auto Calibrate done for DSIC. Set when the auto calibrate sequence for DSI pad bricks is done. Write 1-to-clear.
29	X	MIPI_AUTO_CAL_DONE_DSIB: Debug bit. MIPI Auto Calibrate done for DSIB. Set when the auto calibrate sequence for DSI pad bricks is done. Write 1-to-clear.
28	X	MIPI_AUTO_CAL_DONE_DSIA: Debug bit. MIPI Auto Calibrate done for DSI. Set when the auto calibrate sequence for DSI pad bricks is done. Write 1-to-clear.
24	X	MIPI_AUTO_CAL_DONE_CSIE: Debug bit. MIPI Auto Calibrate done for CSI, set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.
23	X	MIPI_AUTO_CAL_DONE_CSID: MIPI Auto Calibrate done for CSI. Set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.
22	X	MIPI_AUTO_CAL_DONE_CSIC: Debug bit. MIPI Auto Calibrate done for CSI. Set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.
21	X	MIPI_AUTO_CAL_DONE_CSIB: Debug bit. MIPI Auto Calibrate done for CSI. Set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.
20	X	MIPI_AUTO_CAL_DONE_CSIA: Debug bit. MIPI Auto Calibrate done for CSI. Set when the auto calibrate sequence for CSI pad bricks is done. Write 1-to-clear.
16	X	MIPI_AUTO_CAL_DONE: MIPI Auto Calibrate done. Set when the auto-calibrate sequence for MIPI pad bricks is done. Write 1-to-clear.
15:12	X	MIPI_CAL_DRIV_DN_ADJ: Driver Pull-down calibration code generated by MIPI auto calibrate. Valid only after the auto-calibrate sequence has completed (MIPI_CAL_ACTIVE == 0).
11:8	X	MIPI_CAL_DRIV_UP_ADJ: Driver Pull up calibration code generated by MIPI auto calibrate. Valid only after auto calibrate sequence has completed (MIPI_CAL_ACTIVE == 0).
7:4	X	MIPI_CAL_TERMADJ: Termination code generated by MIPI auto calibrate. Valid only after the auto-calibrate sequence has completed (MIPI_CAL_ACTIVE == 0).
0	X	MIPI_CAL_ACTIVE: One when auto calibrate is active.

31.1.4 MIPI_CAL_CILA_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-A MIPI Pads

Offset: 0x14 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEA: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for channel A TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to channel A TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads.
21	0x1	MIPI_CAL_SELA: Select the CSIA pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSA: 2's complement offset for HSPDADJ going to channel A
12:8	0x0	MIPI_CAL_HSPUOSA: 2's complement offset for HSPUADJ going to channel A
4:0	0x0	MIPI_CAL_TERMOSA: 2's complement offset for TERMADJ going to channel A

31.1.5 MIPI_CAL_CILB_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-B MIPI Pads

Offset: 0x18 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEB: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for channel B TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to channel B TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELB: Select the CSIB pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSB: 2's complement offset for HSPDADJ going to channel B
12:8	0x0	MIPI_CAL_HSPUOSB: 2's complement offset for HSPUADJ going to channel B
4:0	0x0	MIPI_CAL_TERMOSB: 2's complement offset for TERMADJ going to channel B

31.1.6 MIPI_CAL_CILC_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-C MIPI Pads

Offset: 0x1c | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEC: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for channel C TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to channel C TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELC: Select the CSIC pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSC: 2's complement offset for HSPDADJ going to channel C
12:8	0x0	MIPI_CAL_HSPUOSC: 2's complement offset for HSPUADJ going to channel C
4:0	0x0	MIPI_CAL_TERMOSC: 2's complement offset for TERMADJ going to channel C

31.1.7 MIPI_CAL_CILD_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-D MIPI Pads

Offset: 0x20 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDED: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for channel D TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to channel D TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELD: Select the CSID PADS for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSD: 2's complement offset for HSPDADJ going to channel D
12:8	0x0	MIPI_CAL_HSPUOSD: 2's complement offset for HSPUADJ going to channel D
4:0	0x0	MIPI_CAL_TERMOSD: 2's complement offset for TERMADJ going to channel D

31.1.8 MIPI_CAL_CILE_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-E MIPI Pads

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bx0xxxxxxxx000000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEE: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for channel E TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to channel E TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x0	MIPI_CAL_SELE: Select the CSIE pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSE: 2's complement offset for HSPDADJ going to channel E
12:8	0x0	MIPI_CAL_HSPUOSE: 2's complement offset for HSPUADJ going to channel E
4:0	0x0	MIPI_CAL_TERMOSE: 2's complement offset for TERMADJ going to channel E

31.1.9 MIPI_CAL_DSIA_MIPI_CAL_CONFIG_0

Calibration Settings for DSIA MIPI Pad

Offset: 0x38 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEDSIA: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELDSIA: Select the DSI pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSDSIA: 2's complement offset for HSPDADJ
12:8	0x0	MIPI_CAL_HSPUOSDSIA: 2's complement offset for HSPUADJ
4:0	0x0	MIPI_CAL_TERMOSDSIA: 2's complement offset for TERMADJ

31.1.10 MIPI_CAL_DSIB_MIPI_CAL_CONFIG_0

Calibration Settings for DSIB MIPI Pad

Offset: 0x3c | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEDSIB: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELDSIB: Select the DSI PADS for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSDSIB: 2's complement offset for HSPDADJ
12:8	0x0	MIPI_CAL_HSPUOSDSIB: 2's complement offset for HSPUADJ
4:0	0x0	MIPI_CAL_TERMOSDSIB: 2's complement offset for TERMADJ

31.1.11 MIPI_CAL_DSIC_MIPI_CAL_CONFIG_0

Calibration Settings for DSIC MIPI Pad

Offset: 0x40 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEDSIC: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELDSIC: Select the DSI pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSDSIC: 2's complement offset for HSPDADJ
12:8	0x0	MIPI_CAL_HSPUOSDSIC: 2's complement offset for HSPUADJ
4:0	0x0	MIPI_CAL_TERMOSDSIC: 2's complement offset for TERMADJ

31.1.12 MIPI_CAL_DSID_MIPI_CAL_CONFIG_0

Calibration Settings for DSID MIPI Pad

Offset: 0x44 | Read/Write: R/W | Reset: 0x00200000 (0bx0xxxxxxx100000xxx00000xxx00000)

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEDSID: When 0 (normal operation), use TERMOS/HSPUOS/HSPDOS as an offset to the Calibration State machine setting for TERMADJ/HSPUADJ/HSPDADJ values to the MIPI pads. When 1, use the register values above as the actual value going to TERMADJ/HSPUADJ/HSPDADJ on the MIPI pads. Writing a one to MIPI_CAL_STARTCAL starts the Calibration State machine.
21	0x1	MIPI_CAL_SELDSID: Select the DSI pads for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSDSID: 2's complement offset for HSPDADJ
12:8	0x0	MIPI_CAL_HSPUOSDSID: 2's complement offset for HSPUADJ
4:0	0x0	MIPI_CAL_TERMOSDSID: 2's complement offset for TERMADJ

31.1.13 MIPI_CAL_MIPI_BIAS_PAD_CFG0_0

MIPI Bias Pad Configuration Register 0

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	MIPI_BIAS_PAD_PDVCLAMP: 1=Power down regulator which supplies current to pre-driver logic.
0	0x0	MIPI_BIAS_PAD_E_VCLAMP_REF: 1=Enable VCLAMP reference voltage

31.1.14 MIPI_CAL_MIPI_BIAS_PAD_CFG1_0

MIPI Bias Pad Configuration Register 1

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxx000xxxxx000xxxxx000xxxxx000)

Bit	Reset	Description
26:24	0x0	PAD_TEST_SEL: Controls which signal to be routed to TEST_OUT 000=VAUXP 001=RUP 010=RDN

Bit	Reset	Description
		011=VREG 1xx=TBD
18:16	0x0	PAD_DRIV_DN_REF: Adjust internal reference level for drive pull-down calibration
10:8	0x0	PAD_DRIV_UP_REF: Adjust internal reference level for drive pull-up calibration
2:0	0x0	PAD_TERM_REF: Adjust internal reference level for termination calibration

31.1.15 MIPI_CAL_MIPI_BIAS_PAD_CFG2_0

MIPI Bias Pad Configuration Register 2

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxx00000000000x000xx10)

Bit	Reset	Description
18:16	0x0	PAD_VCLAMP_LEVEL: VCLAMP level adjustment
15:8	0x0	PAD_SPARE: Spare bit for MIPI Bias Config
6:4	0x0	PAD_VAUXP_LEVEL: VAUXP level adjustment 00 = no adjustment, default 01 = 105% 10 = 110% 11 = 115% 100 = no adjustment 101 = 95% 110 = 90% 111 = 85%
1	0x1	PAD_PDVREG: Power down voltage regulator, 1=power down
0	0x0	PAD_VBYPASS: Bypass bang gap voltage reference

32.0 VIDEO INPUT (VI)

The Video Input block is normally used to receive pixels from an external device, such as a camera, and transfer them to memory. The MIPI CSI interface connects to the VI block and transfers received data to VI.

This section describes the VI registers; however, it is not intended to be a programming guide to VI, as it is expected that NVIDIA supplied drivers will always be used with it. However, the register interface is documented to aid with understanding those drivers.

32.1 VI Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

32.1.1 VI_CTXSW_0

Context Switch Register

Should be common to all modules. Includes the current channel/class (which is writable by software) and the next channel/class (which the hardware sets when it receives a context switch).

Context switch works like this:

Any context switch request triggers an interrupt to the host and causes the new channel/class to be stored in NEXT_CHANNEL/NEXT_CLASS. Software sees that there is a context switch interrupt and does the necessary operations to make the module ready to receive traffic from the new context. It clears the context switch interrupt and writes CURR_CHANNEL/CLASS to the same value as NEXT_CHANNEL/CLASS, which causes a context switch acknowledge packet to be sent to the host. This completes the context switch and allows the host to continue sending data to the module.

Context switches can also be preloaded. If CURR_CLASS/CHANNEL are written and updated to the next CLASS/CHANNEL before the context switch request occurs, an acknowledge will be generated by the module, and no interrupt will be triggered. This is one way for software to avoid dealing with context switch interrupts.

Another way to avoid context switch interrupts is to set the AUTO_ACK bit. This bit tells the module to automatically acknowledge any incoming context switch requests without triggering an interrupt. CURR_* and NEXT_* will be updated by the module so they will always be current.

Offset: 0x20 | Byte Offset: 0x80 | Read/Write: R/W | Reset: 0xXXXXf800 (0bxxxxxxxxxxxxxxxx11111x0000000000)

Bit	R/W	Reset	Description
31:28	RO	X	NEXT_CHANNEL: Next requested channel
25:16	RO	X	NEXT_CLASS: Next requested class
15:12	RW	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	RW	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

32.1.2 VI_INTSTATUS_0

Offset: 0x21 | Byte Offset: 0x84 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	CTXSW_INT: Context switch interrupt status (clear on write)

32.1.3 VI_VI_INPUT_CONTROL_0

VI Input Control

Software must always program parallel cameras in a way that avoids simultaneous HSYNC and VSYNC active edges.

Offset: 0x22 | Byte Offset: 0x88 | Read/Write: R/W | Reset: 0x00000000 (0bx00000000xxxxxxxxxx0000000000000)

Bit	Reset	Description
30	0x0	V_COUNTER: Vertical Counter. 0= Enabled. 1= Disabled (reset to 0) 0 = ENABLED 1 = DISABLED
29	0x0	H_COUNTER: Horizontal Counter. 0= Enabled. 1= Disabled (reset to 0) 0 = ENABLED 1 = DISABLED
28	0x0	FIELD_TYPE: Odd/Even Field type (effective for interlaced video source) 0= Top field is odd field 1= Top field is even field 0 = TOPODD 1 = TOPEVEN
27	0x0	FIELD_DETECT: Reserved.
26:25	0x0	SYNC_FORMAT: Reserved.
24	0x0	VVS_IN_EDGE: VVS input signal active edge which is used as vertical reference of input data. VVS input inversion is evaluated first before determining active edge. 0= Rising edge of VVS is active edge For ITU-R BT.656 data, leading edge of vertical sync is the active edge. 1= Falling edge of VVS is active edge For ITU-R BT.656 data, trailing edge of vertical sync is the active edge. 0 = RISING 1 = FALLING
23	0x0	VHS_IN_EDGE: VHS input signal active edge which is used as horizontal reference of input data. VHS input inversion is evaluated first before determining active edge. 0= Rising edge of VHS is active edge. For ITU-R BT.656 data, leading edge of horizontal sync is the active edge. 1= Falling edge of VHS is active edge For ITU-R BT.656 data, trailing edge of horizontal sync is the active edge. 0 = RISING 1 = FALLING
12:10	0x0	HOST_FORMAT_EXT: Select a data source input to HOST (extension field). (use when input source is host) 000= Source is selected with HOST_FORMAT field (backward compatible) 001= Bayer 10 bpp: 2 16-bit values packed into 32-bit, LSbit aligned {6'b0, bayer, 6'b0, bayer} (to ISP) 010= Bayer 14 bpp: 2 16-bit values packed into 32-bit, LSbit aligned {2'b0, bayer, 2'b0, bayer} (to ISP) 011= RGB565 (to EPP) 100= MSB Alpha + RGB888 (to EPP) 101= MSB Alpha + BGR888 (to EPP) 110= CSI (to CSI) 111= reserved 0 = USE_HOST_FORMAT 1 = BAYER10 2 = BAYER14 3 = RGB565 4 = ARGB8888 5 = ABGR8888 6 = CSI
9:8	0x0	YUV_INPUT_FORMAT: YUV Input Format. This is applicable when the input source is host and the host format is non-planar YUV422. 8 bits per component 00= UYVY => Y1_V0_Y0_U0 MSB to LSB 32bit mapping 01= VYUY => Y1_U0_Y0_V0 10= YUYV => V0_Y1_U0_Y0 11= YVYU => U0_Y1_V0_Y0 0 = UYVY 1 = VYUY

Bit	Reset	Description
		2 = YUYV 3 = YVYU
7:6	0x0	HOST_FORMAT: Host Data Format (effective if input source is host) 00= Non-planar YUV422 (only Y-FIFO is used) 01= Planar YUV420 (Y-FIFO, U-FIFO, V-FIFO are used) 10= Bayer 8-bit - enables ISP 11= Bayer 12-bit - enables ISP 0 = NONPLANAR 1 = PLANAR 2 = BAYER8 3 = BAYER12
5:2	0x0	INPUT_PORT_FORMAT: Reserved.
1	0x0	VIP_INPUT_ENABLE: Reserved.
0	0x0	HOST_INPUT_ENABLE: Host Input Enable 0 = DISABLED 1 = ENABLED

32.1.4 VI_VI_CORE_CONTROL_0

VI Core Control and Output to EPP/ISP

Offset: 0x23 | Byte Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000xxxx0000x0000000)

Bit	R/W	Reset	Description
31	RW	0x0	PLANAR_CONV_INPUT_SEL_EXT: Select source for Planar Conversion Module Input 0= Backwards Compatible - Use PLANAR_CONV_INPUT_SEL setting 1= Use ALT Mux 0 = USE_PLANAR_CONV_INPUT_SEL 1 = YUV422ALT
30	RW	0x0	CSC_INPUT_SEL_EXT: Select source for Color Space Converter 0= Backwards Compatible - Use CSC_INPUT_SEL setting 1= Use ALT Mux 0 = USE_CSC_INPUT_SEL 1 = YUV422ALT
29:27	RW	0x0	INPUT_TO_ALT_MUX: Select a data source from alt mux 001= Host I/F data 010= ISP data, from 444 to 422 converter 011= CSI_PPA data in YUV444NP format 100= CSI_PPA data in YUV422NP format 101= CSI_PPB data in YUV444NP format 110= CSI_PPB data in YUV422NP format 1 = HOST 2 = ISP 3 = CSI_PPA_YUV444 4 = CSI_PPA_YUV422 5 = CSI_PPB_YUV444 6 = CSI_PPB_YUV422
26:24	RW	0x0	INPUT_TO_CORE_EXT: Select a data source input to core (extension field). 000= Source is selected with INPUT_TO_CORE field (backward compatible) 001= CSI_PPA data in YUV444NP format 010= CSI_PPA data in YUV422NP format 011= CSI_PPB data in YUV444NP format 100= CSI_PPB data in YUV422NP format 0 = USE_INPUT_TO_CORE 1 = CSI_PPA_YUV444 2 = CSI_PPA_YUV422 3 = CSI_PPB_YUV444 4 = CSI_PPB_YUV422

Bit	R/W	Reset	Description
23:21	RW	0x0	OUTPUT_TO_ISP_EXT: Select a data source output to ISP (extension field). 000= Source is selected with OUTPUT_TO_ISP field (backward compatible) 001= CSI Pixel Parser A 010= CSI Pixel Parser B 0 = USE_OUTPUT_TO_ISP 1 = CSI_PPA 2 = CSI_PPB
20	WO	0x0	ISP_HOST_STALL_OFF: ISP Host data stall capability is enabled by default. Use this bit to disable the host data stall capability 0= disabled - default allows for VI to turn off the ISP clock to stall the Host. 1= enabled - to turn off the VI's ability to stall the Host when data from ISP comes from Host. 0 = DISABLED 1 = ENABLED
19	RW	0x0	V_DOWNSCALING: Vertical Downscaling (effective if V_AVERAGING is DISABLED) 0= disabled 1= enabled and controlled by V_DOWN_M and V_DOWN_N parameters 0 = DISABLED 1 = ENABLED
18	RW	0x0	V_AVERAGING: Vertical Averaging 0= disabled, V_DOWNSCALING can be used to enable vertical downscaling 1= enabled, V_DOWNSCALING is ignored and vertical downscaling is controlled by V_AVG_FACTOR 0 = DISABLED 1 = ENABLED
17	RW	0x0	H_DOWNSCALING: Horizontal Downscaling (effective if H_AVERAGING is DISABLED) 0= disabled 1= enabled and controlled by H_DOWN_M and H_DOWN_N parameters 0 = DISABLED 1 = ENABLED
16	RW	0x0	H_AVERAGING: Horizontal Averaging 0= disabled, H_DOWNSCALING can be used to enable horizontal downscaling 1= enabled, H_DOWNSCALING is ignored and horizontal downscaling is controlled by H_AVG_FACTOR 0 = DISABLED 1 = ENABLED
11	RW	0x0	CSC_INPUT_SEL: Color Space Conversion Input select 0= YUV422 after down-scaling, POST core 1= YUV422 before downscaling, PRE core 0 = YUV422POST 1 = YUV422PRE
10	RW	0x0	PLANAR_CONV_INPUT_SEL: Planar Conversion Module Input select 0= YUV422 after down-scaling, POST core 1= YUV422 before down-scaling, PRE core 0 = YUV422POST 1 = YUV422PRE
9:8	RW	0x0	INPUT_TO_CORE: Input to VI Core Select between possible data input sources 01= Host I/F data 10= ISP data, from 444 to 422 converter 11= reserved 1 = HOST 2 = ISP
6:5	RW	0x0	ISP_DOWNSAMPLE: Downsample from YUV444 to YUV422 00 = Cosited, take even UV's for each two Y's. 01 = Cosited, take odd UV's for each two Y's. (Not implemented) 10 = Non Cosited, take even U and odd V, use for Bayer pass-thru

Bit	R/W	Reset	Description
			11 = Averaged, average the odd and even UVs. (Not Implemented) 0 = COSITED_EVEN 1 = COSITED_ODD 2 = NONCOSITED 3 = AVERAGED
4:2	RW	0x0	OUTPUT_TO_EPP: Output to EPP enable VI can output a YUV pixel stream to Encoder Pre-Processor (EPP) module 000= Output to EPP is disabled 001= YUV444 stream after downscaling 010= YUV444 stream before downscaling WARNING: FOR YUV444PRE, only the selects in INPUT_TO_CORE are supported. Selects from INPUT_TO_CORE_EXT are not supported since they are duplicated in the CSI* selections of this field. 011= YUV444 stream from ISP, no LPF or downscaling 100= Reserved 101= RGB565,RGB888 from Host 110= CSI_PPA 111= CSI_PPB 0 = DISABLED 1 = YUV444POST 2 = YUV444PRE 3 = YUV444ISP 5 = HOST_RGB 6 = CSI_PPA 7 = CSI_PPB
1:0	RW	0x0	OUTPUT_TO_ISP: Output to ISP Enable data output to ISP 00= Output to ISP is disabled 01= Reserved 10= Host I/F data 11= stereo compositor 0 = DISABLED 2 = HOST 3 = STEREO

32.1.5 VI_VI_FIRST_OUTPUT_CONTROL_0

VI Output Control of YUV/RGB and YUV420P

Offset: 0x24 | Byte Offset: 0x90 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx00000000xxxxxxx0xxxxx000)

Bit	Reset	Description
23	0x0	DESWIZZLE_RAW_PRIME: Enable Deswizzling of Bayer data from ISP This is needed if we need to write processed Bayer data from the middle of ISP pipe. Since ISP data is 'Bayer-packet-as-YUV', This bit enables Deswizzling before writing to memory. The data will be written in the same RAW10 format as CSI_PPA_BAYER/CSI_PPB_BAYER. With this bit disabled, the Raw Prime pixels are written to memory in the format {isp2vi_pixel[7:0],isp2vi_pixel[15:8]} (or {1'b0,ORIGINAL_PIXEL[9:3],1'b0,ORIGINAL_PIXEL[2:0],ORIGINAL_PIXEL[9:6]}) With this bit enabled, the Raw Prime pixels are written to memory in the format {6'd0,isp2vi_pixel[14:13],isp2vi_pixel[12:8],isp2vi_pixel[6:4]} (or {6'd0,ORIGINAL_PIXEL[9:0]}) 0 = DISABLED 1 = ENABLED
22	0x0	OUTPUT_FORMAT_EXT: Extension for Output Format Select 0= Backward compatible - use OUTPUT_FORMAT field 1= Take data from ALT mux 0 = USE_OUTPUT_FORMAT 1 = YUV422ALT
21	0x0	XY_SWAP: Not supported.
20	0x0	V_DIRECTION: V-direction in internal memory
19	0x0	H_DIRECTION: H-direction in internal memory

Bit	Reset	Description
18:17	0x0	YUV_OUTPUT_FORMAT: YUV Output Format. This is applicable when output format is non-planar YUV422. 00= UYVY => Y1_V0_Y1_U0 MSB to LSB 32bit mapping 01= VYUY => Y1_U0_Y1_V0 10= YUYV => V0_Y1_U0_Y0 11= YVYU => U0_Y1_V0_Y0 0 = UYVY 1 = VYUY 2 = YUYV 3 = YVYU
16	0x0	OUTPUT_BYTE_SWAP: Output Byte Swap (effective if input source is host) 0 = DISABLED 1 = ENABLED
8	0x0	LAST_PIXEL_DUPLICATION: For Planar Output Only, enabling this register duplicates the last pixel of each line when the output width is set to an odd number of pixels. Used when JPEGE/MPEGE which requires valid data filled to the word (16-bit) boundary. The Buffer Horizontal Size (Line Stride) must be set to accommodate the extra pixel. Example: Disabled - y0,y1,y2,y3,y4 Enabled - y0,y1,y2,y3,y4,y4 0 = DISABLED 1 = ENABLED
2:0	0x0	OUTPUT_FORMAT: Output data Format Take from the CSC Unit: 000= 16-bit RGB (B5G6R5) 001= 16-bit RGB (B5G6R5) Dithered (This is currently NOT implemented) 010= 24-bit RGB (B8G8R8) Take from the YUV422 Core output path: (Same thing as using YUV422PRE and YUV_SOURCE==CORE_OUTPUT) 011= YUV422 non-planar (U8Y8V8Y8) after downscaling, POST Take from the YUV422 paths: (see YUV_SOURCE field) 100= YUV422 non-planar (U8Y8V8Y8) before downscaling, PRE 101= YUV422 Planar 110= YUV420 Planar 111= YUV420 Planar with Averaging (UV is averaged for each line pair) 0 = RGB16 1 = RGB16D 2 = RGB24 3 = YUV422POST 4 = YUV422PRE 5 = YUV422P 6 = YUV420P 7 = YUV420PA

32.1.6 VI_VI_SECOND_OUTPUT_CONTROL_0

VI Second Output Control of YUV422NP and RGB

Offset: 0x25 | Byte Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx0x000000xxxxxxxxxxxx0000)

Bit	Reset	Description
23	0x0	SECOND_OUTPUT_DESWIZZLE_RAW_PRIME: Enable Deswizzling of Bayer data from ISP. This is needed if we need to write processed Bayer data from the middle of ISP pipe. Since ISP data is 'Bayer-packet-as-YUV', This bit enables Deswizzling before writing to memory. The data will be written in the same RAW10 format as CSI_PPA_BAYER/CSI_PPB_BAYER. With this bit disabled, the Raw Prime pixels are written to memory in the format {isp2vi_pixel[7:0],isp2vi_pixel[15:8]} (or {1'b0,ORIGINAL_PIXEL[9:3],1'b0,ORIGINAL_PIXEL[2:0],ORIGINAL_PIXEL[9:6]}) With this bit enabled, the Raw Prime pixels are written to memory in the format {6'd0,isp2vi_pixel[14:13],isp2vi_pixel[12:8],isp2vi_pixel[6:4]} (or {6'd0,ORIGINAL_PIXEL[9:0]}) 0 = DISABLED 1 = ENABLED
21	0x0	SECOND_XY_SWAP: XY_SWAP IS NO LONGER SUPPORTED
20	0x0	SECOND_V_DIRECTION: Second output's V-direction in internal memory
19	0x0	SECOND_H_DIRECTION: Second output's H-direction in internal memory

Bit	Reset	Description
18:17	0x0	YUV_SECOND_OUTPUT_FORMAT: YUV Second Output Format. This is applicable when output format is non-planar YUV422. 00= UYVY => Y1_V0_Y1_U0 MSB to LSB 32-bit mapping 01= VYUY => Y1_U0_Y1_V0 10= YUYV => V0_Y1_U0_Y0 11= YVYU => U0_Y1_V0_Y0 0 = UYVY 1 = VYUY 2 = YUYV 3 = YVYU
16	0x0	SECOND_OUTPUT_BYTE_SWAP: Output Byte Swap (effective if input source is host) 0 = DISABLED 1 = ENABLED
3:0	0x0	SECOND_OUTPUT_FORMAT: Secondary Output to MC Use case: when VI needs to send decimated preview data and at the same time send non-decimated data to the memory for StretchBLT, meanwhile the StretchBLT is sending EPP stretched data to be encoded. Only YUV422, RGB888, RGB565 are supported. Take from the CSC Unit 0000= 16-bit RGB (B5G6R5), all RGB data can be pre or post decimated depending on mux select programming on the input to the Color Space Converter 0001= 16-bit RGB (B5G6R5) Dithered (This is currently NOT supported) 0010= 24-bit RGB (B8G8R8) Take from the YUV422 Core output path: (Same thing as using YUV422PRE and YUV_SOURCE==CORE_OUTPUT) 0011= YUV422 stream after downscaling, POST Take from the YUV422 paths: (see YUV_SOURCE field) 0100= YUV422 stream before downscaling, PRE Take from the WriteBuffer interface logic, which is used for JPEG Stream 0101= JPEG Stream (Pattern A,B,C) 0110= Reserved 0111= CSI_PPA Bayer direct to memory as a 16-bit value {6'b0, CSI_SVD[15:6]} 1000= CSI_PPB Bayer direct to memory as a 16-bit value {6'b0, CSI_SVD[15:6]} 1010=Reserved 1011=YUV422 stream from the ALT mux 0 = RGB16 1 = RGB16D 2 = RGB24 3 = YUV422POST 4 = YUV422PRE 5 = JPEG_STREAM 7 = CSI_PPA_BAYER 8 = CSI_PPB_BAYER 10 = YUV422ALT

32.1.7 VI_HOST_INPUT_FRAME_SIZE_0

Host Input Frame Width

Input Frame Width and Height give the total input data dimensions. The VI input stage will cull/clip pixels outside the Active Region (see register VI_HOST_H_ACTIVE and VI_HOST_V_ACTIVE). The amount of data per frame is expected to be INPUT_WIDTH * INPUT_HEIGHT * the bytes per pixel (determined from the INPUT_HOST_FORMAT). For planar, the BPP is 1 for the Y FIFO and 1/2 for U and V. For non planar, it is 2.

The Bayer data is treated as 1 byte per pixel, so if it is more, then the input width and the H_ACTIVE should be scaled accordingly, so that internally generated HSYNC and VSYNCs for ISP are correct.

For Bayer input, it is important to insert blanking data for horizontal and vertical, allowing ISP to do sideband calculations.

Offset: 0x26 | Byte Offset: 0x98 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	INPUT_FRAME_HEIGHT: Host Input Frame Height. Specifies in terms of lines the height of the input data coming from host.
12:0	X	INPUT_FRAME_WIDTH: Specifies in terms of pixels the width of the input data coming from host.

32.1.8 VI_HOST_H_ACTIVE_0

VI Horizontal Active

This register defines the horizontal active area of the input video source with respect to the internally generated horizontal sync. (This is for data coming in from host.)

Offset: 0x27 | Byte Offset: 0x9c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	HOST_H_ACTIVE_PERIOD: Horizontal Active Period. This parameter specifies the number of pixels in the horizontal active area. H_ACTIVE_START + H_ACTIVE_PERIOD should be less than 2^NV_VI_H_IN (or 8192). This parameter should be programmed with an even number (bit 16 is ignored internally).
12:0	X	HOST_H_ACTIVE_START: Horizontal Active Start (offset to active). This parameter specifies the number of pixels to be discarded until the first active pixel. If programmed to 0, the first active pixel is the first pixel popped from the Host YUV FIFO.

32.1.9 VI_HOST_V_ACTIVE_0

Vertical Active

This register defines the vertical active area of the input video source with respect to the internally generated vertical sync. (This is for data coming in from the host.)

Offset: 0x28 | Byte Offset: 0xa0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	HOST_V_ACTIVE_PERIOD: Vertical Active Period. This parameter specifies the number of lines in the vertical active area. V_ACTIVE_START + V_ACTIVE_PERIOD should be less than 2^NV_VI_V_IN (or 8192).
12:0	X	HOST_V_ACTIVE_START: Vertical Active Start (offset to active) This parameter specifies the number of horizontal sync active edges from vertical sync active edge to the first vertical active line. If programmed to 0, the first active line starts after the first horizontal sync active edge following the vertical sync active edge.

32.1.10 VI_VIP_H_ACTIVE_0

This register is reserved.

Offset: 0x29 | Byte Offset: 0xa4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	VIP_H_ACTIVE_PERIOD: Reserved.
12:0	X	VIP_H_ACTIVE_START: Reserved.

32.1.11 VI_VIP_V_ACTIVE_0

This register is reserved.

Offset: 0x2a | Byte Offset: 0xa8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	VIP_V_ACTIVE_PERIOD: Reserved.
12:0	X	VIP_V_ACTIVE_START: Reserved.

32.1.12 VI_VI_PEER_CONTROL_0

VI Peer to Peer Control

For all fields in this register:

- 00= Disabled
- 01= First memory
- 10= Second memory
- 11= not defined

Offset: 0x2b | Byte Offset: 0xac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:6	0x0	DISPLAY_B_CONTROL: VI to Display B Control Bus enable. The VI will send a valid buffer signal along with Y,U,V buffer addresses and Frame Start and Frame End 0 = DISABLED 1 = FIRST 2 = SECOND
5:4	0x0	SB_CONTROL: VI to StretchBLT Control Bus enable. The VI will send a valid buffer signal along with buffer index and Frame Start and Frame End The VI to SB control bus is separate from the VI to JPEGE/MPEGE bus. This control bus is controlled by the "2nd Output to MC" write client interface. 0 = DISABLED 1 = FIRST 2 = SECOND
3:2	0x0	ENCODER_CONTROL: VI to JPEGE and MPEGE Control Bus enable. VI will send a valid buffer signal along with buffer index and Frame Start and Frame End 0 = DISABLED 1 = FIRST 2 = SECOND
1:0	0x0	DISPLAY_CONTROL: VI to Display Control Bus enable. The VI will send a valid buffer signal along with Y,U,V buffer addresses and Frame Start and Frame End 0 = DISABLED 1 = FIRST 2 = SECOND

32.1.13 VI_VI_DMA_SELECT_0

Host DMA select

Offset: 0x2c | Byte Offset: 0xb0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1:0	0x0	DMA_REQUEST: Host DMA Request enable at end of block. Request to host DMA can be enabled every time a block of video input data is written to memory. 00= Disabled 01= Write Buffer DMA for RAW data stream 10= First memory 11= Second memory 0 = DISABLED 1 = STREAM 2 = FIRST 3 = SECOND

32.1.14 VI_HOST_DMA_WRITE_BUFFER_0

Host DMA Write Buffer Configuration Registers

Offset: 0x2d | Byte Offset: 0xb4 | Read/Write: R/W | Reset: 0x0XXXXXXX (0bxxxx000xxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
27:26	0x0	SOURCE_SEL: Data source selection 01= CSI_PPA 10= CSI_PPB 1 = CSI_PPA 2 = CSI_PPB
25	0x0	DMA_ENABLE: DMA Enable 0 = DISABLED 1 = ENABLED
24:16	X	BUFFER_NUMBER: Buffer Number
15:0	X	BUFFER_SIZE: Buffer Size

32.1.15 VI_HOST_DMA_BASE_ADDRESS_0

Host DMA Write Buffer Configuration Registers

Offset: 0x2e | Byte Offset: 0xb8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DMA_BASE_ADDR: Base Address

32.1.16 VI_HOST_DMA_WRITE_BUFFER_STATUS_0

Host DMA Write Buffer Status Register

Offset: 0x2f | Byte Offset: 0xbc | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26:0	X	WB_STATUS: Status of the Host DMA write buffer

32.1.17 VI_HOST_DMA_WRITE_PEND_BUFCOUNT_0

Host DMA Write Buffer Pending Buffer Count

Offset: 0x30 | Byte Offset: 0xc0 | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8:0	X	PEND_BUFCOUNT: Pending buffer count of the Host DMA write buffer.

32.1.18 VI_VB0_START_ADDRESS_FIRST_0

Video Buffer O Start Address for First Output

First Output Registers

These registers are used to setup the first of two memory outputs for VI Address Y, U, V; Frame size; Count; Size (line stride and block height); and Buffer Stride

Offset: 0x31 | Byte Offset: 0xc4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VB0_START_ADDRESS_1: This is the byte address of video buffer 0 if output data format is RGB or YUV non-planar. This is the byte address of video buffer 0 Y-plane if output data format is YUV planar.

32.1.19 VI_VB0_BASE_ADDRESS_FIRST_0

Video Buffer O BASE Address for First Output

BASE address is used in Tiling mode. The BASE address always points to the left_upper corner of a surface. A surface can contain multiple buffers, in circular_buffer case.

Writes to the BASE address register will cause the corresponding internal buffer index to be reset to zero.

Offset: 0x32 | Byte Offset: 0xc8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VB0_BASE_ADDRESS_1: This is the first byte address of video buffer 0. This is the byte address of video buffer 0 Y-plane if output data format is planar.

32.1.20 VI_VB0_START_ADDRESS_U_0

Video Buffer O Start Address U (linked to First Output)

Offset: 0x33 | Byte Offset: 0xcc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VB0_START_ADDRESS_U: This is the byte address of video buffer 0 U-plane, if output data format is YUV planar. Due to clock gating, the primary OUTPUT_TO_MEMORY must be enabled and the OUTPUT_FORMAT must be set to a planar format prior to writing this register

32.1.21 VI_VB0_BASE_ADDRESS_U_0

Video Buffer O BASE Address U (linked to First Output)

Offset: 0x34 | Byte Offset: 0xd0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VB0_BASE_ADDRESS_U: This is the first byte address of video buffer 0 U-plane, if output data format is planar.

32.1.22 VI_VB0_START_ADDRESS_V_0

Video Buffer O Start Address V (linked to First Output)

Offset: 0x35 | Byte Offset: 0xd4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VB0_START_ADDRESS_V: This is the byte address of video buffer 0 V-plane, if output data format is YUV planar. Due to clock gating, the primary OUTPUT_TO_MEMORY must be enabled and the OUTPUT_FORMAT must be set to a planar format prior to writing this register

32.1.23 VI_VB0_BASE_ADDRESS_V_0

Video Buffer O BASE Address V (linked to First Output)

Offset: 0x36 | Byte Offset: 0xd8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VB0_BASE_ADDRESS_V: This is the byte address of video buffer 0 V-plane, if output data format is YUV planar.

32.1.24 VI_VB0_SCRATCH_ADDRESS_UV_0

Video Buffer O Scratch Address UV (linked to First Output)

Offset: 0x37 | Byte Offset: 0xdc | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VB0_SCRATCH_ADDRESS_UV: If OUTPUT_FORMAT is YUV420PA, this is used. This is the byte address of video buffer 0. UV intermediate data is saved here during the YUV422 to YUV420PA conversion. The size allocated needs to match the FIRST_FRAME_WIDTH register setting.

32.1.25 VI_FIRST_OUTPUT_FRAME_SIZE_0

Width and height of first output frame

This is the size of the frame being written to memory. Apply decimation or averaging to calculate the output frame size. Whether or not downscaling is used specify whatever the size of the frame being written to memory.

Offset: 0x38 | Byte Offset: 0xe0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	FIRST_FRAME_HEIGHT: Frame height in lines which the VI needs to process
12:0	X	FIRST_FRAME_WIDTH: Frame width in pixel which the VI needs to process

32.1.26 VI_VB0_COUNT_FIRST_0

Video Buffer Set 0 Count for First Output

Offset: 0x39 | Byte Offset: 0xe4 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VB0_COUNT_1: Video Buffer Set 0 Count. This specifies the number of buffers in video buffer set 0.

32.1.27 VI_VB0_SIZE_FIRST_0

Video Buffer Set 0 Size for First Output

Offset: 0x3a | Byte Offset: 0xe8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	VB0_V_SIZE_1: Video Buffer Set 0 Vertical Size. This specifies the number of lines in each buffer in video buffer set 0. This value will be divided by 2 for chroma buffers for YUV420 planar formats
12:0	X	VB0_H_SIZE_1: Video Buffer Set 0 Horizontal Size. This parameter specifies the line stride (in pixels) for lines in the video buffer set 0. For YUV non-planar format, this parameter must be programmed as multiple of 2 pixels (bit 0 is ignored). For YUV planar format, this parameter must be programmed as multiple of 8 pixels (bits 2-0 are ignored) and it specifies the luma line stride or twice the chroma line stride. This value is divided by 2 for chroma buffers for YUV422 and YUV420 planar formats

32.1.28 VI_VB0_BUFFER_STRIDE_FIRST_0

Video Buffer Set 0 Buffer Stride

Offset: 0x3b | Byte Offset: 0xec | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:30	X	VB0_BUFFER_STRIDE_C: Video Buffer Set 0 Chroma Buffer Stride 00= Equal to Luma Buffer Stride 01= Equal to Luma Buffer Stride divided by 2 in this case Luma Buffer Stride should be multiple of 2 bytes. 10= Equal to Luma Buffer Stride divided by 4 in this case Luma Buffer Stride should be multiple of 4 bytes. 1x= Reserved 0 = CBS1X 1 = CBS2X 2 = CBS4X
29:0	X	VB0_BUFFER_STRIDE_L: Video Buffer Set 0 Luma Buffer Stride This is luma buffer stride (in bytes)

32.1.29 VI_VB0_START_ADDRESS_SECOND_0

Video Buffer O Start Address for Second Output

Second Output Registers

These registers are used to set up the second of two memory outputs for VI Address; Frame size; Count; Size (line stride and block height); and Buffer Stride.

Offset: 0x3c | Byte Offset: 0xf0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VB0_START_ADDRESS_2: This is byte address of video buffer 0 if output data format is RGB or YUV non-planar. This is byte address of video buffer 0. This output data is read by the SB function of the GR2D block.

32.1.30 VI_VB0_BASE_ADDRESS_SECOND_0

Video Buffer O Base Address for Second Output

Offset: 0x3d | Byte Offset: 0xf4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	VB0_BASE_ADDRESS_2: This is the byte address of video buffer 0, if output data format is RGB or non-planar. This is the first byte address of the video buffer

32.1.31 VI_SECOND_OUTPUT_FRAME_SIZE_0

Width and Height of Second Output Frame

Offset: 0x3e | Byte Offset: 0xf8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	SECOND_FRAME_HEIGHT: frame height in lines which the VI needs to process
12:0	X	SECOND_FRAME_WIDTH: frame width in pixel which the VI needs to process

32.1.32 VI_VB0_COUNT_SECOND_0

Video Buffer Set 0 Count for Second Output

Offset: 0x3f | Byte Offset: 0xfc | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	VB0_COUNT_2: This specifies the number of buffers in video buffer set 0.

32.1.33 VI_VB0_SIZE_SECOND_0

Video Buffer Set 0 Size for Second Output

Offset: 0x40 | Byte Offset: 0x100 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	VB0_V_SIZE_2: Video Buffer Set 0 Vertical Size. This specifies the number of lines in each buffer in video buffer set 0.
12:0	X	VB0_H_SIZE_2: Video Buffer Set 0 Horizontal Size. This parameter specifies the line stride (in pixels) for lines in the video buffer set 0. For YUV non-planar format, this parameter must be programmed as multiple of 2 pixels (bit 0 is ignored).

32.1.34 VI_VB0_BUFFER_STRIDE_SECOND_0

Video Buffer Set 0 Buffer Stride for Second Output

Offset: 0x41 | Byte Offset: 0x104 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:0	X	VB0_BUFFER_STRIDE_2: Video Buffer Set 0 Luma Buffer Stride. This is luma buffer stride (in bytes)

32.1.35 VI_H_LPF_CONTROL_0

VI Horizontal Low-Pass Filter (LPF) Control

Offset: 0x42 | Byte Offset: 0x108 | Read/Write: R/W | Reset: 0x02400240 (0bxxx0001001000000xxx0001001000000)

Bit	Reset	Description
28:16	0x240	H_LPF_C: Horizontal LPF Chrominance filter. This controls low-pass filter for U V data.
12:0	0x240	H_LPF_L: Horizontal LPF Luminance filter. This controls low-pass filter for Y data.

32.1.36 VI_H_DOWNSCALE_CONTROL_0

VI Horizontal Downscaling Control

Horizontal pixel processing starts with horizontal low-pass filtering.

Following horizontal low-pass filtering, horizontal downscaling (decimation) can then be performed with or without horizontal averaging.

If horizontal downscaling (decimation) is performed without horizontal averaging, the down-scaling factor is specified by input active period and output frame size.

If horizontal downscaling is performed with horizontal averaging, the downscaling factors are limited to few factors determined by H_AVG_CONTROL. When enabling averaging PLEASE be careful that the input and output ratios match the formula for the averaging decimation ratio exactly to the pixel/line.

The formulae for each of the Averaging Decimation Ratio are as follows:

- x = input size
- y(x) = output size
- 2-pixel averaging and 1/2 downscaling: y(x) = Floor(x/2)
- 4-pixel averaging and 1/3 downscaling: y(x) = Floor((x-1)/3)
- 4-pixel averaging and 1/4 downscaling: y(x) = Floor(x/4)
- 8-pixel averaging and 1/7 downscaling: y(x) = Floor((x-1)/7)
- 8-pixel averaging and 1/8 downscaling: y(x) = Floor(x/8)

Horizontal Decimation Algorithm

The Horizontal Decimator decides which pixels to drop by using a simple DDA algorithm. The accumulator will continue to add the value of the output width (numerator) for each pixel until the sum is equal to or greater than the input width (denominator). When the sum is greater than or equal to the input width (denominator), the hardware will flag that pixel as a pixel to be written out to memory. At the same time, the input width (denominator) will be subtracted from the sum, and the difference will be loaded back into the accumulator for the next line. By default the accumulator is initialized with 0's upon reset. However, the user can set the H_DEC_INIT_VAL to initialize the accumulator with a certain value from 0 to the input width (denominator). Any H_DEC_INIT_VAL that is greater or equal to the difference of the input width (denominator) and the output width (numerator) will cause the first pixel to be written out to memory. This register shifts the phase of the decimation pattern.

Offset: 0x43 | Byte Offset: 0x10c | Read/Write: R/W | Reset: 0x00000X0X (0bxxx0000000000000xxxxxxxxxxxx00xx)

Bit	Reset	Description
28:16	0x0	H_DEC_INIT_VAL: Horizontal Decimation Accumulator Initial Value. The user may initialize the H-Dec accumulator with a value between 0-(H_ACTIVE_PERIOD) to change the phase of the decimation pattern. This will allow the user to decide which is the first pixel to keep.
10:8	X	H_AVG_CONTROL: Horizontal Averaging Control. This specifies the number of pixels to average and to decimate horizontally. 000= 2-pixel averaging and 1/2 downscaling 001= 4-pixel averaging and 1/3 downscaling 010= 4-pixel averaging and 1/4 downscaling 011= 8-pixel averaging and 1/7 downscaling 100= 8-pixel averaging and 1/8 downscaling other= reserved 0 = A2D2 1 = A4D3 2 = A4D4 3 = A8D7 4 = A8D8

Bit	Reset	Description
3:2	0x0	INPUT_H_SIZE_SEL_EXT: Selects input horizontal size into scalers (extension field) 00= Horizontal size selected with INPUT_H_SIZE_SEL field (backward compatible) 01= Horizontal size of CSI_PPA is provided by CSI_PPA_H_ACTIVE register 10= Horizontal size of CSI_PPB is provided by CSI_PPB_H_ACTIVE register 11= Horizontal size of ISP is provided by ISP_H_ACTIVE register 0 = USE_INPUT_H_SIZE_SEL 1 = CSI_PPA 2 = CSI_PPB 3 = ISP
1	X	OUTPUT_H_SIZE_SEL: Output Horizontal Size Select. Selects between the first and second memory output frame widths for the numerator in the downscaling ratio. Uses FIRST_FRAME_WIDTH or SECOND_FRAME_WIDTH. This is effective only when H_AVERAGING is DISABLED and H_DOWNSCALING is ENABLED. 0 = FIRST 1 = SECOND
0	X	INPUT_H_SIZE_SEL: Input Horizontal Size Select. The VIP option is deprecated. Only the HOST option is valid. Selects the HOST input active area width for the denominator in the downscaling ratio. Uses HOST_H_ACTIVE_PERIOD, which is the width of the data after cropping. This is effective only when H_AVERAGING is DISABLED and H_DOWNSCALING is ENABLED. 1 = HOST

32.1.37 VI_V_DOWNSCALE_CONTROL_0

VI Vertical Downscaling Control

Vertical processing consists of optional vertical downscaling (decimation) which can be performed with or without vertical averaging.

If vertical downscaling (decimation) is performed without vertical averaging, the down-scaling factor is specified by input active period and output frame size.

If horizontal down-scaling is performed with vertical averaging, the downscaling factors are limited to few factors determined by V_AVG_CONTROL. When enabling averaging, you must ensure that the input and output ratios match the formula for the averaging decimation ratio exactly to the pixel/line.

The formulae for each of the Averaging Decimation Ratio are as follows:

- x = input size
- y(x) = output size
- 2-pixel averaging and 1/2 downscaling: y(x) = Floor(x/2)
- 4-pixel averaging and 1/3 downscaling: y(x) = Floor((x-1)/3)
- 4-pixel averaging and 1/4 downscaling: y(x) = Floor(x/4)
- 8-pixel averaging and 1/7 downscaling: y(x) = Floor((x-1)/7)
- 8-pixel averaging and 1/8 downscaling: y(x) = Floor(x/8)

Vertical Decimation Algorithm (same as the Horizontal Decimation Algorithm)

The Vertical Decimator decides which pixels to drop by using a simple DDA algorithm. The accumulator will continue to add the value of the output height (numerator) for each line until the sum is equal to or greater than the input height (denominator). When the sum is greater than or equal to the input height (denominator), the hardware will flag that line as a line to be written out to memory. At the same time, the input height (denominator) will be subtracted from the sum, and the difference will be loaded back into the accumulator for the next line. By default the accumulator is initialized with 0's upon reset. However, the user can set the V_DEC_INIT_VAL to initialize the accumulator with a certain value from 0 to the input height (denominator). Any V_DEC_INIT_VAL that is greater or equal to the difference of the input height (denominator), and the output height (numerator) will cause the first line to be written out to memory. This register shifts the phase of the decimation pattern.

Offset: 0x44 | Byte Offset: 0x110 | Read/Write: R/W | Reset: 0x0000XX0X (0bxxx0000000000000xxxxxxxxxxxx00xx)

Bit	Reset	Description
28	X	IGNORE_FIELD: Specifies whether odd/even field affects vertical decimation. 0 = disabled - odd/even field affects the vertical downscaling 1 = enabled - field is ignored in vertical downscaling 0 = DISABLED 1 = ENABLED
28:16	0x0	V_DEC_INIT_VAL: Vertical Decimation Accumulator Initial Value The user may initialize the V-Dec accumulator with a value between 0-(V_ACTIVE_PERIOD) to change the phase of the decimation pattern. This will allow the user to decide which is the first line to keep.
13	X	MULTI_TAP_V_AVG_FILTER: Multi-Tap Vertical Averaging Filter 0 = disabled 1 = enabled. This will enable the Multi-Tap filtering when the Vertical Averaging is enabled. The filter settings will depend on the V_AVG_CONTROL value. 000 - 3 Taps (1,2,1)/4 001 - 5 Taps (1,2,2,2,1)/8 010 - 6 Taps (1,1,2,2,1,1)/8 011 - 11 Taps (1,1,1,2,2,2,2,2,1,1,1)/16 100 - 12 Taps (1,1,1,1,2,2,2,2,1,1,1,1)/16 0 = DISABLED 1 = ENABLED
12	X	FLEXIBLE_VSCALE: Flexible Vertical Scaling 0 = disabled, V_AVG_CONTROL specifies both vertical averaging and downscaling factor. 1 = enabled, fixed 2-line averaging with vertical downscaling controlled by V_DOWN_N and V_DOWN_D. 0 = DISABLED 1 = ENABLED
10:8	X	V_AVG_CONTROL: Vertical Averaging Control. This specifies the number of lines to average and to decimate vertically. 000= 2-line averaging and 1/2 downscaling 001= 4-line averaging and 1/3 downscaling 010= 4-line averaging and 1/4 downscaling 011= 8-line averaging and 1/7 downscaling 100= 8-line averaging and 1/8 downscaling other= reserved 0 = A2D2 1 = A4D3 2 = A4D4 3 = A8D7 4 = A8D8
3:2	0x0	INPUT_V_SIZE_SEL_EXT: Selects input vertical size into scalers (extension field) 00= Vert. size selected with INPUT_V_SIZE_SEL field (backward compatible) 01= Vert. size of CSI_PPA is provided by CSI_PPA_V_ACTIVE register 10= Vert. size of CSI_PPB is provided by CSI_PPB_V_ACTIVE register 11= Vert. size of ISP is provided by ISP_V_ACTIVE register 0 = USE_INPUT_V_SIZE_SEL 1 = CSI_PPA 2 = CSI_PPB 3 = ISP
1	X	OUTPUT_V_SIZE_SEL: Output Vertical Size Select. Selects between the first and second memory output frame heights for the numerator in the downscaling ratio. Uses FIRST_FRAME_HEIGHT or SECOND_FRAME_HEIGHT. This is effective only when V_AVERAGING is DISABLED and V_DOWNSCALING is ENABLED. 0 = FIRST 1 = SECOND
0	X	INPUT_V_SIZE_SEL: Input Vertical Size Select. The VIP option is deprecated. Only the HOST option is valid. Selects the HOST input active area height for the denominator in the downscaling ratio. Uses HOST_V_ACTIVE_PERIOD, which is the height of the data after cropping. This is effective only when V_AVERAGING is DISABLED and V_DOWNSCALING is ENABLED. 1 = HOST

32.1.38 VI_CSC_Y_0

CSC Y Offset and Gain

Color Space Conversion coefficients.

The CSC can be used for YUV to RGB conversion with brightness and hue/saturation control.

For Y color, the Y offset is applied first and saturation (clipping) is performed immediately after the Y offset is applied.

R = sat(KYRGB * sat(Y + YOF) + KUR * U + KVR * V)

G = sat(KYRGB * sat(Y + YOF) + KUG * U + KVG * V)

B = sat(KYRGB * sat(Y + YOF) + KUB * U + KVB * V)

Saturation and rounding is performed in the range of 0 to 255 for the above equations.

Typical values are:

YOF = -16.000, KYRGB = 1.1644

KUR = 0.0000, KVR = -1.5960

KUG = -0.3918, KVG = -0.8130

KUB = 2.0172, KVB = 0.0000

KUR and KVB are typically 0.0000 but they may be programmed non-zero for hue rotation.

The CSC can also take RGB input, in which case YOF, KVB, KUG, KUR should be programmed to 0, and KYRGB will be forced to 0 by the hardware for generating R and B. KYRGB will not be forced to 0 for generating G. KVR, KYRGB, and KUB can be programmed to 1.0 or used as gain control for R, G, B correspondingly.

Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

Offset: 0x45 | Byte Offset: 0x114 | Read/Write: R/W | Reset: 0x0XXX00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	KYRGB: Y Gain for R, G, B colors in 2.8 format
7:0	X	YOF: Y Offset in s.7.0 format

32.1.39 VI_CSC_UV_R_0

CSC U & V coefficient for R

Offset: 0x46 | Byte Offset: 0x118 | Read/Write: R/W | Reset: 0x0XXX0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26:16	X	KVR: V coefficients for R in s.2.8 format
10:0	X	KUR: U coefficients for R in s.2.8 format

32.1.40 VI_CSC_UV_G_0

CSC U & V coefficient for G

Offset: 0x47 | Byte Offset: 0x11c | Read/Write: R/W | Reset: 0x0XXX0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25:16	X	KVG: V coefficients for G in s.1.8 format
9:0	X	KUG: U coefficients for G in s.1.8 format

32.1.41 VI_CSC_UV_B_0

CSC U & V coefficient for B

Offset: 0x48 | Byte Offset: 0x120 | Read/Write: R/W | Reset: 0x0XXX0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26:16	X	KVB: V coefficients for B in s.2.8 format
10:0	X	KUB: U coefficients for B in s.2.8 format

32.1.42 VI_CSC_ALPHA_0

RGB Color Space Converter Alpha value

Offset: 0x49 | Byte Offset: 0x124 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	RGB888_ALPHA: When output format to memory is selected for RGB888, the pixel data is 32-bit aligned. The value programmed here will be appended to the RGB888 data as the 8 MSBs and can be used as an alpha value.

32.1.43 VI_HOST_VSYNC_0

Valid when INPUT_SOURCE is HOST

Offset: 0x4a | Byte Offset: 0x128 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	HOST_VSYNC_TRIGGER: This triggers VI's internal VSYNC generation. Always write once to this register with '1' before writing the Frame's data to Y_FIFO_DATA

32.1.44 VI_COMMAND_0

VI Command

This register is used initialize VI module when INPUT_SOURCE is HOST.

This register has a dual use purpose. Host input VSYNC is created by writing to this register.

Offset: 0x4b | Byte Offset: 0x12c | Read/Write: R/W | Reset: 0xXXXX0X00 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
28:16	X	V_COUNTER_THRESHOLD: Vertical Counter Threshold. This specifies a threshold which, when exceeded, would generate the vertical counter interrupt if the interrupt is enabled. This is used to detect the case when the host is sending too many input data than expected by VI module.
11:8	X	Y_FIFO_THRESHOLD: Y-FIFO Threshold. This specifies maximum number of filled locations in Y-FIFO for the Y-FIFO Threshold Status bit.
0	0x0	PROCESS_FIELD: Process Odd/Even field (effective when INPUT_SOURCE is HOST). Writing to this bit will initialize VI to receive one field of video. 0= odd field 1= even field 0 = ODD 1 = EVEN

32.1.45 VI_HOST_FIFO_STATUS_0

Host FIFO status

This is not needed if host input video goes through command buffer interface.

Offset: 0x4c | Byte Offset: 0x130 | Read/Write: RO | Reset: 0x0000XX0X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:12	X	V_FIFO_STATUS: This indicates the number of filled locations in V-FIFO. If the returned value is 3'h0, the FIFO is empty and if the returned value is 3'h7 then the FIFO is full.
10:8	X	U_FIFO_STATUS: This indicates the number of filled locations in U-FIFO. If the returned value is 3'h0, the FIFO is empty and if the returned value is 3'h7 then the FIFO is full.
3:0	X	Y_FIFO_STATUS: This indicates the number of filled locations in Y-FIFO. If the returned value is 4'h0, the FIFO is empty and if the returned value is 4'hF then the FIFO is full.

32.1.46 VI_INTERRUPT_MASK_0

Interrupt Mask

Offset: 0x4d | Byte Offset: 0x134 | Read/Write: R/W | Reset: 0x00000000 (0bxxx00000000x00000000000000000000)

Bit	Reset	Description
28	0x0	SECOND_OUTPUT_DROP_MC_DATA_INT_MASK: This bit controls interrupt when VI drops data to MC. 0 = DISABLED 1 = ENABLED
27	0x0	FIRST_OUTPUT_DROP_MC_DATA_INT_MASK: This bit controls interrupt when VI drops data to MC. 0 = DISABLED 1 = ENABLED
26	0x0	TS_OTHER_PROTOCOL_ERROR_INT_MASK: This bit controls interrupt when the ISDB-T input get a packet which means FEC+BODY in total size but FEC and BODY do not match FEC_SIZE and BODY_SIZE 0 = DISABLED 1 = ENABLED
25	0x0	TS_OVERRUN_ERROR_INT_MASK: This bit controls interrupt when the ISDB-T input get an overrun error 0 = DISABLED 1 = ENABLED
24	0x0	TS_UNDERRUN_ERROR_INT_MASK: This bit controls interrupt when the ISDB-T input get an underrun error 0 = DISABLED 1 = ENABLED
23	0x0	TS_UPSTREAM_ERROR_INT_MASK: This bit controls interrupt when the ISDB-T VI input gets an upstream error. 0 = DISABLED 1 = ENABLED
22	0x0	RAISE_STREAM_2_INT_MASK: Stream 2 raise. This bit controls interrupt when the the Stream 2 Raise is enabled and returned 0 = DISABLED 1 = ENABLED
21	0x0	RAISE_STREAM_1_INT_MASK: Stream 1 raise. This bit controls interrupt when the the Stream 1 Raise is enabled and returned 0 = DISABLED 1 = ENABLED
19	0x0	DMA_STALL_INT_MASK: Write Buffer DMA to VI. Stalls VI and causes an error This bit controls interrupt when the VI drops raw 8-bit stream data because the Write Buffer DMA is stalling. 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
18	0x0	SECOND_OUTPUT_PEER_STALL_INT_MASK: VI to Peer stall - Second Memory Output This bit controls interrupt when the VI drops peer bus packet(s) because the peer is stalling the second output peer bus and data is coming from the pins 0 = DISABLED 1 = ENABLED
17	0x0	FIRST_OUTPUT_PEER_STALL_INT_MASK: VI to Peer stall - First Memory Output. This bit controls interrupt when the VI drops peer bus packet(s) because the peer is stalling the first output peer bus and data is coming from the pins 0 = DISABLED 1 = ENABLED
16	0x0	YUV420PA_ERROR_INT_MASK: YUV420PA Error Interrupt Mask. This bit controls interrupt when the VI does not average data because the line buffer data is not ready from the memory controller. The VI will write unaveraged data and will write the U,V data from the even line in such cases. 0 = DISABLED 1 = ENABLED
15	0x0	EPP_ERROR_INT_MASK: VI to EPP Error Interrupt Mask. This bit controls interrupt when the VI drops data to the EPP because the EPP is stalling the vi2epp bus and data is coming from the pins 0 = DISABLED 1 = ENABLED
14	0x0	FRAME_SECOND_OUTPUT_INT_MASK: Buffer Done Second Output Interrupt Mask. This bit controls interrupt when the Second Output to memory has written a frame to memory. 0 = DISABLED 1 = ENABLED
13	0x0	BUFFER_SECOND_OUTPUT_INT_MASK: Buffer Done Second Output Interrupt Mask This bit controls interrupt when the Second Output to memory has written a buffer to memory. 0 = DISABLED 1 = ENABLED
12	0x0	FRAME_FIRST_OUTPUT_INT_MASK: Frame Done First Output Interrupt Mask This bit controls interrupt when the First Output to memory has written a frame to memory. 0 = DISABLED 1 = ENABLED
11	0x0	BUFFER_FIRST_OUTPUT_INT_MASK: Buffer Done First Output Interrupt Mask This bit controls interrupt when the First Output to memory has written a buffer to memory. 0 = DISABLED 1 = ENABLED
10	0x0	Y_THRESHOLD_INT_MASK: Y-FIFO Threshold Interrupt Mask This bit controls interrupt when the number of filled locations in Y-FIFO is equal or greater than the Y_FIFO_THRESHOLD value. This bit should be set to 1 only when INPUT_SOURCE is HOST. 0 = DISABLED 1 = ENABLED
9	0x0	V_COUNTER_INT_MASK: Vertical Counter Interrupt Mask (effective when VIDEO_SOURCE is HOST) This bit controls interrupt when the vertical counter threshold is reached. 0 = DISABLED 1 = ENABLED
8	0x0	VVS_INT_MASK: VVS pin Interrupt Mask This bit controls interrupt when VVS rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
7	0x0	VHS_INT_MASK: VHS pin Interrupt Mask This bit controls interrupt when VHS rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
6	0x0	VGP6_INT_MASK: VGP6 pin Interrupt Mask This bit controls interrupt when VGP6 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
5	0x0	VGP5_INT_MASK: VGP5 pin Interrupt Mask This bit controls interrupt when VGP5 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
4	0x0	VGP4_INT_MASK: VGP4 pin Interrupt Mask. This bit controls interrupt when VGP4 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
3	0x0	VD11_INT_MASK: VD11 pin Interrupt Mask. This bit controls interrupt when VD11 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
2	0x0	VD10_INT_MASK: VD10 pin Interrupt Mask. This bit controls interrupt when VD10 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
1	0x0	VD9_INT_MASK: VD9 pin Interrupt Mask. This bit controls interrupt when VD9 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED
0	0x0	VD8_INT_MASK: VD8 pin Interrupt Mask. This bit controls interrupt when VD8 rising/falling edge is detected. 0 = DISABLED 1 = ENABLED

32.1.47 VI_INTERRUPT_TYPE_SELECT_0

Interrupt Type Select

Offset: 0x4e | Byte Offset: 0x138 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	VVS_INT_TYPE: VVS pin Interrupt Type. This bit controls interrupt VVS 0 = EDGE 1 = LEVEL
7	0x0	VHS_INT_TYPE: VHS pin Interrupt Type. This bit controls interrupt VHS 0 = EDGE 1 = LEVEL
6	0x0	VGP6_INT_TYPE: VGP6 pin Interrupt Type. This bit controls interrupt VGP6 0 = EDGE 1 = LEVEL
5	0x0	VGP5_INT_TYPE: VGP5 pin Interrupt Type. This bit controls interrupt VGP5 0 = EDGE 1 = LEVEL
4	0x0	VGP4_INT_TYPE: VGP4 pin Interrupt Type. This bit controls interrupt VGP4 0 = EDGE 1 = LEVEL
3	0x0	VD11_INT_TYPE: VD11 pin Interrupt Type. This bit controls interrupt VD11 0 = EDGE 1 = LEVEL
2	0x0	VD10_INT_TYPE: VD10 pin Interrupt Type. This bit controls interrupt VD10 0 = EDGE 1 = LEVEL
1	0x0	VD9_INT_TYPE: VD9 pin Interrupt Type. This bit controls interrupt VD9 0 = EDGE 1 = LEVEL

Bit	Reset	Description
0	0x0	VD8_INT_TYPE: VD8 pin Interrupt Type. This bit controls interrupt VD8 if edge or level type 0 = EDGE 1 = LEVEL

32.1.48 VI_INTERRUPT_POLARITY_SELECT_0

Interrupt Polarity Select

Offset: 0x4f | Byte Offset: 0x13c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx000000000)

Bit	Reset	Description
8	0x0	VVS_INT_POLARITY: VVS pin Interrupt Type. This bit controls interrupt VVS 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
7	0x0	VHS_INT_POLARITY: VHS pin Interrupt Type. This bit controls interrupt VHS 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
6	0x0	VGP6_INT_POLARITY: VGP6 pin Interrupt Type. This bit controls interrupt VGP6 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
5	0x0	VGP5_INT_POLARITY: VGP5 pin Interrupt Type. This bit controls interrupt VGP5 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
4	0x0	VGP4_INT_POLARITY: VGP4 pin Interrupt Type. This bit controls interrupt VGP4 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
3	0x0	VD11_INT_POLARITY: VD11 pin Interrupt Type. This bit controls interrupt VD11 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
2	0x0	VD10_INT_POLARITY: VD10 pin Interrupt Type. This bit controls interrupt VD10 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
1	0x0	VD9_INT_POLARITY: VD9 pin Interrupt Type. This bit controls interrupt VD9 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH
0	0x0	VD8_INT_POLARITY: VD8 pin Interrupt Type. This bit controls interrupt VD8 if edge or level type. 0= falling edge or low level 1= rising edge or high level 0 = LOW 1 = HIGH

32.1.49 VI_INTERRUPT_STATUS_0

Interrupt Enable

This register returns interrupt status when read. Except for bits 15-14, when this register is written, the interrupt status corresponding to the bits written with 1 will be reset.

Interrupt status corresponding to the bits written with 0 will be left unchanged.

Note that interrupt status bits can be set even when their corresponding interrupt enable bits, in VI10R, are cleared. When these bits are set and their corresponding interrupt enable bits are set, an interrupt is generated. The interrupt can be cleared, or left unchanged, by writing 1 or 0, respectively, to the corresponding bits in this register.

Clearing the interrupt status bits does not affect the interrupt enable bits.

Offset: 0x50 | Byte Offset: 0x140 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28	X	SECOND_OUTPUT_DROP_MC_DATA_INT_STATUS: If SECOND_OUTPUT is dropping data to MC, INTR will be set. 0 = NOINTR 1 = INTR
27	X	FIRST_OUTPUT_DROP_MC_DATA_INT_STATUS: If FIRST_OUTPUT is dropping data to MC, INTR will be set. 0 = NOINTR 1 = INTR
26	X	TS_OTHER_PROTOCOL_ERROR_INT_STATUS: This bit shows the status of the condition when the ISDB-T input gets another protocol error (ex: total packet received is FEC_SIZE+BODY_SIZE but the individual FEC portion != FEC_SIZE and the individual BODY portion != BODY_SIZE 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
25	X	TS_OVERRUN_ERROR_INT_STATUS: This bit shows the status of the condition when the ISDB-T input gets an overrun error (more bytes in packet than specified 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
24	X	TS_UNDERRUN_ERROR_INT_STATUS: This bit shows the status of the condition when the ISDB-T input gets an underrun error (START condition detected prior to receiving a full packet) 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
23	X	TS_UPSTREAM_ERROR_INT_STATUS: This bit shows the status of the condition when the ISDB-T VI input gets an upstream error (error from the tuner) 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
22	X	RAISE_STREAM_2_INT_STATUS: This bit shows the status of the condition when the Raise Stream 2 returns to the Host 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR

Bit	Reset	Description
21	X	RAISE_STREAM_1_INT_STATUS: This bit shows the status of the condition when the Raise Stream 1 returns to the Host 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
20	X	FIELD_STATUS: Top or Bottom Field Status This bit specifies whether the last received video data field is top field or bottom field as defined by FIELD_TYPE bit. This bit cannot be reset by software by writing a 1. 0= Bottom field received 1= Top field received 0 = BOTTOM 1 = TOP
19	X	DMA_STALL_INT_STATUS: This bit shows the status of the condition when the VI drops data to the Write Buffer DMA 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
18	X	SECOND_OUTPUT_PEER_STALL_INT_STATUS: This bit shows the status of if the VI dropped a buffer packet to the peer communicating with the second memory output 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
17	X	FIRST_OUTPUT_PEER_STALL_INT_STATUS: This bit shows the status of if the VI dropped a buffer packet to the peer communicating with the first memory output 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
16	X	YUV420PA_ERROR_INT_STATUS: YUV420PA Error Interrupt Enable This bit shows the status of if the VI does not average data because the line buffer data is not ready from the memory controller. The VI will write unaveraged data and will write the U,V data from the even line in such cases. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
15	X	EPP_ERROR_INT_STATUS: VI to EPP Error Interrupt Enable This bit controls interrupt when the VI drops data to the EPP because the EPP is stalling the vi2epp bus and data is coming from the pins 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
14	X	FRAME_SECOND_OUTPUT_INT_STATUS: Frame Done Second Output Interrupt Status. This bit is set when a frame has been written to memory by the second output. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
13	X	BUFFER_SECOND_OUTPUT_INT_STATUS: Buffer Done Second Output Interrupt Status. This bit is set when a buffer has been written to memory by the second output. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
12	X	FRAME_FIRST_OUTPUT_INT_STATUS: Frame Done First Output Interrupt Status. This bit is set when a frame has been written to memory by the first output. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR

Bit	Reset	Description
11	X	BUFFER_FIRST_OUTPUT_INT_STATUS: Buffer Done First Output Interrupt Status. This bit is set when a buffer has been written to memory by the first output. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
10	X	Y_THRESHOLD_INT_STATUS: Y-FIFO Threshold Interrupt Enable. This bit controls interrupt when the number of filled locations in Y-FIFO is equal or greater than the Y_FIFO_THRESHOLD value. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
9	X	V_COUNTER_INT_STATUS: Vertical Counter Interrupt Status (effective when VIDEO_SOURCE is HOST). This bit controls interrupt when the vertical counter threshold is reached. 0 = NOINTR 1 = INTR
8	X	VVS_INT_STATUS: VVS pin Interrupt Status. This bit controls interrupt when VVS rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
7	X	VHS_INT_STATUS: VHS pin Interrupt Status. This bit controls interrupt when VHS rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
6	X	VGP6_INT_STATUS: VGP6 pin Interrupt Status. This bit controls interrupt when VGP6 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
5	X	VGP5_INT_STATUS: VGP5 pin Interrupt Status. This bit controls interrupt when VGP5 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
4	X	VGP4_INT_STATUS: VGP4 pin Interrupt Status. This bit controls interrupt when VGP4 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
3	X	VD11_INT_STATUS: VD11 pin Interrupt Status. This bit controls interrupt when VD11 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
2	X	VD10_INT_STATUS: VD10 pin Interrupt Status. This bit controls interrupt when VD10 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR
1	X	VD9_INT_STATUS: VD9 pin Interrupt Status. This bit controls interrupt when VD9 rising/falling edge is detected. 0= Interrupt not detected

Bit	Reset	Description
		1= Interrupt detected 0 = NOINTR 1 = INTR
0	X	VD8_INT_STATUS: VD8 pin Interrupt Status. This bit controls interrupt when VD8 rising/falling edge is detected. 0= Interrupt not detected 1= Interrupt detected 0 = NOINTR 1 = INTR

32.1.50 VI_VIP_INPUT_STATUS_0

This register is reserved.

Offset: 0x51 | Byte Offset: 0x144 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	FRAME_COUNT
15:0	X	LINE_COUNT

32.1.51 VI_VIDEO_BUFFER_STATUS_0

Interrupt Enable

Offset: 0x52 | Byte Offset: 0x148 | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAW_STREAM_WRITE_COUNT: Write count of the Raw Stream Write FIFO. This is the FIFO used to synchronize the data coming from pads into the VI clock domain.
15:8	X	SECOND_VIDEO_BUFFER_STATUS: Buffer status. This specifies the buffer number of the last video data field written to memory
7:0	X	FIRST_VIDEO_BUFFER_STATUS: Buffer status. This specifies the buffer number of the last video data field written to memory

32.1.52 VI_SYNC_OUTPUT_0

VI H and V Sync Output control

This register is reserved.

Offset: 0x53 | Byte Offset: 0x14c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:19	X	VVS_OUTPUT_PERIOD
18:16	X	VVS_OUTPUT_WIDTH
15:3	X	VHS_OUTPUT_PERIOD
2:0	X	VHS_OUTPUT_WIDTH

32.1.53 VI_VVS_OUTPUT_DELAY_0

VI V Sync Output Delay

Offset: 0x54 | Byte Offset: 0x150 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
3:0	X	VVS_OUTPUT_DELAY: This specifies the number of VI clock cycles from leading edge of VHS to leading edge of VVS. Programmed value is actual value + 2 so valid value ranges from -2 to 13.

32.1.54 VI_PWM_CONTROL_0

VI Pulse Width Modulation Control

The PWM feature has been deprecated.This register is reserved.

Offset: 0x55 | Byte Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0b00000000xx00xxxx00000000xxx0xxx0)

Bit	Reset	Description
31:0	0x0	Reserved.

32.1.55 VI_PWM_SELECT_PULSE_A_0

PWM Pulse Select A

The next 4 registers select which of the internal 128 pulses to be output. Each bit in the four registers corresponds to one internal pulse.

Offset: 0x56 | Byte Offset: 0x158 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_A: PWM Select bits 31 to 0

32.1.56 VI_PWM_SELECT_PULSE_B_0

PWM Pulse Select B

Offset: 0x57 | Byte Offset: 0x15c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_B: PWM Select bits 63 to 32

32.1.57 VI_PWM_SELECT_PULSE_C_0

PWM Pulse Select C

Offset: 0x58 | Byte Offset: 0x160 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_C: PWM Select bits 95 to 64

32.1.58 VI_PWM_SELECT_PULSE_D_0

PWM Pulse Select D

Offset: 0x59 | Byte Offset: 0x164 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_D: PWM Select bits 127 to 96

32.1.59 VI_VI_DATA_INPUT_CONTROL_0

VI Input Mask

Offset: 0x5a | Byte Offset: 0x168 | Read/Write: R/W | Reset: 0x00000fff (0bxxxxxxxxxxxxxxxxxxxx111111111111)

Bit	Reset	Description
11:0	0xfff	VI_DATA_INPUT_MASK: Mask the VD[11:0] pin inputs to the VI core and ISP. The mask is not applied to the Host GPIO read value

32.1.60 VI_PIN_INPUT_ENABLE_0

VI pins Input Enable

Offset: 0x5b | Byte Offset: 0x16c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx000000x00x000000000000)

Bit	Reset	Description
21	0x0	VGP6_INPUT_ENABLE: VGP6 pin Input Enable This bit controls VGP6 pin input. 0 = DISABLED 1 = ENABLED
20	0x0	VGP5_INPUT_ENABLE: VGP5 pin Input Enable This bit controls VGP5 pin input. 0 = DISABLED 1 = ENABLED
19	0x0	VGP4_INPUT_ENABLE: VGP4 pin Input Enable This bit controls VGP4 pin input. 0 = DISABLED 1 = ENABLED
18	0x0	VGP3_INPUT_ENABLE: VGP3 pin Input Enable This bit controls VGP3 pin input. 0 = DISABLED 1 = ENABLED
17	0x0	VGP2_INPUT_ENABLE: VGP2 pin Input Enable This bit controls VGP2 pin input. 0 = DISABLED 1 = ENABLED
16	0x0	VGP1_INPUT_ENABLE: VGP1 pin Input Enable This bit controls VGP1 pin input. 0 = DISABLED 1 = ENABLED
14	0x0	VVS_INPUT_ENABLE: VVS pin Input Enable This bit controls VVS pin input. 0 = DISABLED 1 = ENABLED
13	0x0	VHS_INPUT_ENABLE: VHS pin Input Enable This bit controls VHS pin input. 0 = DISABLED 1 = ENABLED
11	0x0	VD11_INPUT_ENABLE: VD11 pin Input Enable This bit controls VD7 pin input. 0 = DISABLED 1 = ENABLED
10	0x0	VD10_INPUT_ENABLE: VD10 pin Input Enable This bit controls VD7 pin input. 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
9	0x0	VD9_INPUT_ENABLE: VD9 pin Input Enable This bit controls VD7 pin input. 0 = DISABLED 1 = ENABLED
8	0x0	VD8_INPUT_ENABLE: VD8 pin Input Enable This bit controls VD7 pin input. 0 = DISABLED 1 = ENABLED
7	0x0	VD7_INPUT_ENABLE: VD7 pin Input Enable This bit controls VD7 pin input. 0 = DISABLED 1 = ENABLED
6	0x0	VD6_INPUT_ENABLE: VD6 pin Input Enable This bit controls VD6 pin input. 0 = DISABLED 1 = ENABLED
5	0x0	VD5_INPUT_ENABLE: VD5 pin Input Enable This bit controls VD5 pin input. 0 = DISABLED 1 = ENABLED
4	0x0	VD4_INPUT_ENABLE: VD4 pin Input Enable This bit controls VD4 pin input. 0 = DISABLED 1 = ENABLED
3	0x0	VD3_INPUT_ENABLE: VD3 pin Input Enable This bit controls VD3 pin input. 0 = DISABLED 1 = ENABLED
2	0x0	VD2_INPUT_ENABLE: VD2 pin Input Enable This bit controls VD2 pin input. 0 = DISABLED 1 = ENABLED
1	0x0	VD1_INPUT_ENABLE: VD1 pin Input Enable This bit controls VD1 pin input. 0= Disabled 1= Enabled 0 = DISABLED 1 = ENABLED
0	0x0	VD0_INPUT_ENABLE: VD0 pin Input Enable This bit controls VD0 pin input. 0 = DISABLED 1 = ENABLED

32.1.61 VI_PIN_OUTPUT_ENABLE_0

VI pins Output Enable

Offset: 0x5c | Byte Offset: 0x170 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxx000000x000000000000000)

Bit	Reset	Description
21	0x0	VGP6_OUTPUT_ENABLE: VGP6 pin Output Enable. This bit controls VGP6 pin output. 0 = DISABLED 1 = ENABLED
20	0x0	VGP5_OUTPUT_ENABLE: VGP5 pin Output Enable. This bit controls VGP5 pin output. 0 = DISABLED 1 = ENABLED
19	0x0	VGP4_OUTPUT_ENABLE: VGP4 pin Output Enable. This bit controls VGP4 pin output. 0 = DISABLED 1 = ENABLED
18	0x0	VGP3_OUTPUT_ENABLE: VGP3 pin Output Enable. This bit controls VGP3 pin output. 0 = DISABLED 1 = ENABLED
17	0x0	VGP2_OUTPUT_ENABLE: VGP2 pin Output Enable. This bit controls VGP2 pin output. 0 = DISABLED 1 = ENABLED

Bit	Reset	Description
16	0x0	VGP1_OUTPUT_ENABLE: VGP1 pin Output Enable. This bit controls VGP1 pin output. 0 = DISABLED 1 = ENABLED
14	0x0	VVS_OUTPUT_ENABLE: VVS pin Output Enable. This bit controls VVS pin output. 0 = DISABLED 1 = ENABLED
13	0x0	VHS_OUTPUT_ENABLE: VHS pin Output Enable. This bit controls VHS pin output. 0 = DISABLED 1 = ENABLED
12	0x0	VSCK_OUTPUT_ENABLE: VSCK pin Output Enable. This bit controls VSCK pin output. 0 = DISABLED 1 = ENABLED
11	0x0	VD11_OUTPUT_ENABLE: VD11 pin Output Enable. This bit controls VD7 pin output. 0 = DISABLED 1 = ENABLED
10	0x0	VD10_OUTPUT_ENABLE: VD10 pin Output Enable. This bit controls VD7 pin output. 0 = DISABLED 1 = ENABLED
9	0x0	VD9_OUTPUT_ENABLE: VD9 pin Output Enable. This bit controls VD7 pin output. 0 = DISABLED 1 = ENABLED
8	0x0	VD8_OUTPUT_ENABLE: VD8 pin Output Enable. This bit controls VD7 pin output. 0 = DISABLED 1 = ENABLED
7	0x0	VD7_OUTPUT_ENABLE: VD7 pin Output Enable. This bit controls VD7 pin output. 0 = DISABLED 1 = ENABLED
6	0x0	VD6_OUTPUT_ENABLE: VD6 pin Output Enable. This bit controls VD6 pin output. 0 = DISABLED 1 = ENABLED
5	0x0	VD5_OUTPUT_ENABLE: VD5 pin Output Enable This bit controls VD5 pin output. 0 = DISABLED 1 = ENABLED
4	0x0	VD4_OUTPUT_ENABLE: VD4 pin Output Enable This bit controls VD4 pin output. 0 = DISABLED 1 = ENABLED
3	0x0	VD3_OUTPUT_ENABLE: VD3 pin Output Enable This bit controls VD3 pin output. 0 = DISABLED 1 = ENABLED
2	0x0	VD2_OUTPUT_ENABLE: VD2 pin Output Enable This bit controls VD2 pin output. 0 = DISABLED 1 = ENABLED
1	0x0	VD1_OUTPUT_ENABLE: VD1 pin Output Enable This bit controls VD1 pin output. 0 = DISABLED 1 = ENABLED
0	0x0	VD0_OUTPUT_ENABLE: VD0 pin Output Enable This bit controls VD0 pin output. 0 = DISABLED 1 = ENABLED

32.1.62 VI_PIN_INVERSION_0

VI Pins Input/Output Inversion 0 reserved

Offset: 0x5d | Byte Offset: 0x174 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx000xxxxxxxxxxxxx00x)

Bit	Reset	Description
18	0x0	VVS_OUT_INVERSION: VVS pin Output Inversion 0= VVS output is not inverted (VVS output is active high) 1= VVS output is inverted (VVS output is active low) 0 = DISABLED 1 = ENABLED
17	0x0	VHS_OUT_INVERSION: VHS pin Output Inversion 0= VHS output is not inverted (VHS output is active high) 1= VHS output is inverted (VHS output is active low) 0 = DISABLED 1 = ENABLED
16	0x0	VSCK_OUT_INVERSION: VSCK pin Output Inversion 0= VSCK output is not inverted 1= VSCK output is inverted 0 = DISABLED 1 = ENABLED
2	0x0	VVS_IN_INVERSION: VVS pin Input Inversion 0= VVS input is not inverted (VVS input is active high) 1= VVS input is inverted (VVS input is active low) 0 = DISABLED 1 = ENABLED
1	0x0	VHS_IN_INVERSION: VHS pin Input Inversion 0= VHS input is not inverted (VHS input is active high) 1= VHS input is inverted (VHS input is active low) 0 = DISABLED 1 = ENABLED

32.1.63 VI_PIN_INPUT_DATA_0

VI pins Input Data

This register contains input data when the video camera interface pins are used as general-purpose input pins. The pin data read from this register is not affected by the pin input inversion bits.

Offset: 0x5e | Byte Offset: 0x178 | Read/Write: RO | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
21	X	VGP6_INPUT_DATA: VGP6 pin Input Data (effective if VGP6_INPUT_ENABLE is ENABLED) 0= VGP6 input low 1= VGP6 input high
20	X	VGP5_INPUT_DATA: VGP5 pin Input Data (effective if VGP5_INPUT_ENABLE is ENABLED) 0= VGP5 input low 1= VGP5 input high
19	X	VGP4_INPUT_DATA: VGP4 pin Input Data (effective if VGP4_INPUT_ENABLE is ENABLED) 0= VGP4 input low 1= VGP4 input high
18	X	VGP3_INPUT_DATA: VGP3 pin Input Data (effective if VGP3_INPUT_ENABLE is ENABLED) 0= VGP3 input low 1= VGP3 input high
17	X	VGP2_INPUT_DATA: VGP2 pin Input Data (effective if VGP2_INPUT_ENABLE is ENABLED) 0= VGP2 input low 1= VGP2 input high

Bit	Reset	Description
16	X	VGP1_INPUT_DATA: VGP1 pin Input Data (effective if VGP1_INPUT_ENABLE is ENABLED) 0= VGP1 input low 1= VGP1 input high
14	X	VVS_INPUT_DATA: VVS pin Input Data (effective if VVS_INPUT_ENABLE is ENABLED) 0= VVS input low 1= VVS input high
13	X	VHS_INPUT_DATA: VHS pin Input Data (effective if VHS_INPUT_ENABLE is ENABLED) 0= VHS input low 1= VHS input high
11	X	VD11_INPUT_DATA: VD11 pin Input Data (effective if VD11_INPUT_ENABLE is ENABLED) 0= VD11 input low 1= VD11 input high
10	X	VD10_INPUT_DATA: VD10 pin Input Data (effective if VD10_INPUT_ENABLE is ENABLED) 0= VD10 input low 1= VD10 input high
9	X	VD9_INPUT_DATA: VD9 pin Input Data (effective if VD9_INPUT_ENABLE is ENABLED) 0= VD9 input low 1= VD9 input high
8	X	VD8_INPUT_DATA: VD8 pin Input Data (effective if VD8_INPUT_ENABLE is ENABLED) 0= VD8 input low 1= VD8 input high
7	X	VD7_INPUT_DATA: VD7 pin Input Data (effective if VD7_INPUT_ENABLE is ENABLED) 0= VD7 input low 1= VD7 input high
6	X	VD6_INPUT_DATA: VD6 pin Input Data (effective if VD6_INPUT_ENABLE is ENABLED) 0= VD6 input low 1= VD6 input high
5	X	VD5_INPUT_DATA: VD5 pin Input Data (effective if VD5_INPUT_ENABLE is ENABLED) 0= VD5 input low 1= VD5 input high
4	X	VD4_INPUT_DATA: VD4 pin Input Data (effective if VD4_INPUT_ENABLE is ENABLED) 0= VD4 input low 1= VD4 input high
3	X	VD3_INPUT_DATA: VD3 pin Input Data (effective if VD3_INPUT_ENABLE is ENABLED) 0= VD3 input low 1= VD3 input high
2	X	VD2_INPUT_DATA: VD2 pin Input Data (effective if VD2_INPUT_ENABLE is ENABLED) 0= VD2 input low 1= VD2 input high
1	X	VD1_INPUT_DATA: VD1 pin Input Data (effective if VD1_INPUT_ENABLE is ENABLED) 0= VD1 input low 1= VD1 input high
0	X	VD0_INPUT_DATA: VD0 pin Input Data (effective if VD0_INPUT_ENABLE is ENABLED) 0= VD0 input low 1= VD0 input high

32.1.64 VI_PIN_OUTPUT_DATA_0

VI Pins Output Data

This register contains output data when the video camera interface pins are used as general-purpose output pins. When a bit in this register is written, the data bits can be output on the corresponding pin if the corresponding pin output buffer is enabled, and the pin output control select bits are programmed to output the bit in this register.

The output signal at the pin is affected by the corresponding pin output inversion bit.

Offset: 0x5f | Byte Offset: 0x17c | Read/Write: R/W | Reset: 0x00XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
21	X	VGP6_OUTPUT_DATA: VGP6 pin Output Data (effective if VGP6_OUTPUT_ENABLE is ENABLED and VGP6_OUTPUT_SELECT is DATA)
20	X	VGP5_OUTPUT_DATA: VGP5 pin Output Data (effective if VGP5_OUTPUT_ENABLE is ENABLED and VGP5_OUTPUT_SELECT is DATA)
19	X	VGP4_OUTPUT_DATA: VGP4 pin Output Data (effective if VGP4_OUTPUT_ENABLE is ENABLED and VGP4_OUTPUT_SELECT is DATA)
18	X	VGP3_OUTPUT_DATA: VGP3 pin Output Data (effective if VGP3_OUTPUT_ENABLE is ENABLED and VGP3_OUTPUT_SELECT is DATA)
17	X	VGP2_OUTPUT_DATA: VGP2 pin Output Data (effective if VGP2_OUTPUT_ENABLE is ENABLED and VGP2_OUTPUT_SELECT is DATA)
16	X	VGP1_OUTPUT_DATA: VGP1 pin Output Data (effective if VGP1_OUTPUT_ENABLE is ENABLED and VGP1_OUTPUT_SELECT is DATA)
14	X	VVS_OUTPUT_DATA: VVS pin Output Data (effective if VVS_OUTPUT_ENABLE is ENABLED and VVS_OUTPUT_SELECT is DATA)
13	X	VHS_OUTPUT_DATA: VHS pin Output Data (effective if VHS_OUTPUT_ENABLE is ENABLED and VHS_OUTPUT_SELECT is DATA)
12	X	VSCK_OUTPUT_DATA: VSCK pin Output Data (effective if VSCK_OUTPUT_ENABLE is ENABLED and VSCK_OUTPUT_SELECT is DATA)
11	X	VD11_OUTPUT_DATA: VD11 pin Output Data (effective if VD11_OUTPUT_ENABLE is ENABLED and VD11_OUTPUT_SELECT is DATA)
10	X	VD10_OUTPUT_DATA: VD10 pin Output Data (effective if VD10_OUTPUT_ENABLE is ENABLED and VD10_OUTPUT_SELECT is DATA)
9	X	VD9_OUTPUT_DATA: VD9 pin Output Data (effective if VD9_OUTPUT_ENABLE is ENABLED and VD9_OUTPUT_SELECT is DATA)
8	X	VD8_OUTPUT_DATA: VD8 pin Output Data (effective if VD8_OUTPUT_ENABLE is ENABLED and VD8_OUTPUT_SELECT is DATA)
7	X	VD7_OUTPUT_DATA: VD7 pin Output Data (effective if VD7_OUTPUT_ENABLE is ENABLED and VD7_OUTPUT_SELECT is DATA)
6	X	VD6_OUTPUT_DATA: VD6 pin Output Data (effective if VD6_OUTPUT_ENABLE is ENABLED and VD6_OUTPUT_SELECT is DATA)
5	X	VD5_OUTPUT_DATA: VD5 pin Output Data (effective if VD5_OUTPUT_ENABLE is ENABLED and VD5_OUTPUT_SELECT is DATA)
4	X	VD4_OUTPUT_DATA: VD4 pin Output Data (effective if VD4_OUTPUT_ENABLE is ENABLED and VD4_OUTPUT_SELECT is DATA)
3	X	VD3_OUTPUT_DATA: VD3 pin Output Data (effective if VD3_OUTPUT_ENABLE is ENABLED and VD3_OUTPUT_SELECT is DATA)
2	X	VD2_OUTPUT_DATA: VD2 pin Output Data (effective if VD2_OUTPUT_ENABLE is ENABLED and VD2_OUTPUT_SELECT is DATA)
1	X	VD1_OUTPUT_DATA: VD1 pin Output Data (effective if VD1_OUTPUT_ENABLE is ENABLED and VD1_OUTPUT_SELECT is DATA)

Bit	Reset	Description
0	X	VD0_OUTPUT_DATA: VD0 pin Output Data (effective if VD0_OUTPUT_ENABLE is ENABLED and VD0_OUTPUT_SELECT is DATA)

32.1.65 VI_PIN_OUTPUT_SELECT_0

VI pins Output Select

This is the mux select used at the Pad Macro. For VCLK, VHSYNC, VVSYNC. Selects between the register programmed GPIO outputs (set to 0) and the internally generated VICLK, HSYNC, VSYNC (set to 1). For VGP1-VGP2, selects between the I2C outputs (set to 0) and the VI register programmed GPIO outputs (set to 1). For VD0-VD11, reserved for future use data pins output will be driven by GPIO outputs if enabled.

Offset: 0x60 | Byte Offset: 0x180 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx000000x000000000000000)

Bit	Reset	Description
21	0x0	PIN_OUTPUT_SELECT_vgp6: Pin Output Select VGP6 0= select VGP6 register data out 1= select PWM out 0 = DATA 1 = PWM
20	0x0	PIN_OUTPUT_SELECT_vgp5: Pin Output Select VGP5 0 = VGP5 output register 1 = 1'b0
19	0x0	PIN_OUTPUT_SELECT_vgp4: Pin Output Select VGP4 0 = VGP4 output register 1 = 1'b0
18	0x0	PIN_OUTPUT_SELECT_vgp3: Pin Output Select VGP3 0 = VGP3 output register 1 = 1'b0
17	0x0	PIN_OUTPUT_SELECT_vgp2: Pin Output Select VGP2 0 = I^2C SDA pin 1 = 1'b0
16	0x0	PIN_OUTPUT_SELECT_vgp1: Pin Output Select VGP1 0 = I^2C SCK pin 1 = 1'b0
14	0x0	PIN_OUTPUT_SELECT_vvs: Pin Output Select VVSYNC
13	0x0	PIN_OUTPUT_SELECT_vhs: Pin Output Select VHSYNC
12	0x0	PIN_OUTPUT_SELECT_vclk: Pin Output Select VCLK
11	0x0	PIN_OUTPUT_SELECT_vd11: Pin Output Select VD11
10	0x0	PIN_OUTPUT_SELECT_vd10: Pin Output Select VD10
9	0x0	PIN_OUTPUT_SELECT_vd9: Pin Output Select VD9
8	0x0	PIN_OUTPUT_SELECT_vd8: Pin Output Select VD8
7	0x0	PIN_OUTPUT_SELECT_vd7: Pin Output Select VD7
6	0x0	PIN_OUTPUT_SELECT_vd6: Pin Output Select VD6
5	0x0	PIN_OUTPUT_SELECT_vd5: Pin Output Select VD5
4	0x0	PIN_OUTPUT_SELECT_vd4: Pin Output Select VD4
3	0x0	PIN_OUTPUT_SELECT_vd3: Pin Output Select VD3
2	0x0	PIN_OUTPUT_SELECT_vd2: Pin Output Select VD2
1	0x0	PIN_OUTPUT_SELECT_vd1: Pin Output Select VD1

Bit	Reset	Description
0	0x0	PIN_OUTPUT_SELECT_vd0: Pin Output Select VD0

32.1.66 VI_RAISE_VIP_BUFFER_FIRST_OUTPUT_0

This register is reserved.

A raise written when decimation or averaging is selected in the VI is not supported.

Offset: 0x61 | Byte Offset: 0x184 | Read/Write: R/W | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_BUFFER_1_CHANNEL
4:0	RW	X	RAISE_BUFFER_1_VECTOR

32.1.67 VI_RAISE_VIP_FRAME_FIRST_OUTPUT_0

This register is reserved.

Offset: 0x62 | Byte Offset: 0x188 | Read/Write: R/W | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_FRAME_1_CHANNEL
4:0	RW	X	RAISE_FRAME_1_VECTOR

32.1.68 VI_RAISE_VIP_BUFFER_SECOND_OUTPUT_0

This register is reserved.

Offset: 0x63 | Byte Offset: 0x18c | Read/Write: R/W | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_BUFFER_2_CHANNEL
4:0	RW	X	RAISE_BUFFER_2_VECTOR

32.1.69 VI_RAISE_VIP_FRAME_SECOND_OUTPUT_0

This register is reserved.

Offset: 0x64 | Byte Offset: 0x190 | Read/Write: R/W | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_FRAME_2_CHANNEL
4:0	RW	X	RAISE_FRAME_2_VECTOR

32.1.70 VI_RAISE_HOST_FIRST_OUTPUT_0

Raise Vector When from Host

Offset: 0x65 | Byte Offset: 0x194 | Read/Write: R/W | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_HOST_1_CHANNEL
4:0	RW	X	RAISE_HOST_1_VECTOR

32.1.71 VI_RAISE_HOST_SECOND_OUTPUT_0

Raise Vector When from Host

Offset: 0x66 | Byte Offset: 0x198 | Read/Write: R/W | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_HOST_2_CHANNEL
4:0	RW	X	RAISE_HOST_2_VECTOR

32.1.72 VI_RAISE_EPP_0

Raise Vector at Line End

The EPP receives the raise request via the Simple Stream Video Data bus

This raise needs to be written during the horizontal blanking period. (After end of line.)

This register is only valid if the input source is host.

Offset: 0x67 | Byte Offset: 0x19c | Read/Write: R/W | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_EPP_CHANNEL
4:0	RW	X	RAISE_EPP_VECTOR

32.1.73 VI_CAMERA_CONTROL_0

VI camera control bits

Software must always program parallel cameras in a way that avoids simultaneous HSYNC and VSYNC active edges.

Offset: 0x68 | Byte Offset: 0x1a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000)

Bit	R/W	Reset	Description
2	RW	0x0	STOP_CAPTURE: Disables camera capturing VI_ENABLE after the next end of frame. 0 = DISABLED 1 = ENABLED
1	RW	0x0	TEST_MODE_ENABLE: Test Mode Enable 0 = DISABLED 1 = ENABLED
0	RO	0x0	VIP_ENABLE: Reserved.

32.1.74 VI_VI_ENABLE_0

VI Enables

Offset: 0x69 | Byte Offset: 0x1a4 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx01)

Bit	Reset	Description
1	0x0	SW_FLOW_CONTROL_OUT1: Software enable flow control for output1 0 = DISABLE 1 = ENABLE
0	0x1	FIRST_OUTPUT_TO_MEMORY: First Output to Memory 0 = ENABLED 1 = DISABLED

32.1.75 VI_VI_ENABLE_2_0

VI Enables second output

Offset: 0x6a | Byte Offset: 0x1a8 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx01)

Bit	Reset	Description
1	0x0	SW_FLOW_CONTROL_OUT2: Software enable flow control for output2 0 = DISABLE 1 = ENABLE
0	0x1	SECOND_OUTPUT_TO_MEMORY: Second Output to Memory Disabling output to memory may be set if only output to encoder pre-processor is needed. This will also power down all logic which is only used to send output data to memory. 0 = ENABLED 1 = DISABLED

32.1.76 VI_VI_RAISE_0

VI Enables second output

Offset: 0x6b | Byte Offset: 0x1ac | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	RAISE_ON_EDGE: Makes Raises edge triggered not level sensitive, i.e., only return raise at the end of frame, not in the middle of the V-blank time. 0 = DISABLED 1 = ENABLED

32.1.77 VI_Y_FIFO_WRITE_0

YUV 4:2:0 Planar Y-FIFO, YUV 4:2:2 non-Planar YUV FIFO

Host YUV FIFO offsets. This register space is used for Host Video Data writes.

YUV 4:2:0 planar for re-encoding as well as YUV 4:2:2 data

Offset: 0x6c | Byte Offset: 0x1b0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	Y_FIFO_DATA

32.1.78 VI_U_FIFO_WRITE_0

YUV 4:2:0 Planar U-FIFO

Offset: 0x6d | Byte Offset: 0x1b4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	U_FIFO_DATA

32.1.79 VI_V_FIFO_WRITE_0

YUV 4:2:0 Planar V-FIFO

Offset: 0x6e | Byte Offset: 0x1b8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	V_FIFO_DATA

32.1.80 VI_VI_MCCIF_FIFOCTRL_0

Memory Client Interface FIFO Control Register

> **Note:** The FIFO timing aspects of this register are not supported but are retained for software compatibility.

The clock enable fields of this register control the second-level clock gating for the MC side of the MCCIF.

A '1' written to the override field will result in that clock reverting to the legacy mode of operation (where the msc*clk is on whenever the client clock is enabled.

Offset: 0x6f | Byte Offset: 0x1bc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx00xxxxxxxxxxxxxx00)

Bit	Reset	Description
17	0x0	VI_RCLK_OVERRIDE
16	0x0	VI_WCLK_OVERRIDE
1	DISABLE	VI_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	VI_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

32.1.81 VI_TIMEOUT_WCOAL_VI_0

Memory Client Interface FIFO Control Register

> **Note:** The FIFO timing aspects of this register are not supported but are retained for software compatibility.

The clock enable fields of this register control the second-level clock gating for the MC side of the MCCIF.

A '1' written to the override field will result in that clock reverting to the legacy mode of operation (where the msc*clk is on whenever the client clock is enabled.

Write Coalescing Time-Out Register

> **Note:** Write coalescing is not supported by the MCCIF clients.
> Registers are retained for backwards compatibility but are not used by the hardware.

Offset: 0x70 | Byte Offset: 0x1c0 | Read/Write: R/W | Reset: 0x32323232 (0b00110010000110010000110010000110010)

Bit	Reset	Description
31:24	0x32	VIWY_WCOAL_TMVAL
23:16	0x32	VIWV_WCOAL_TMVAL
15:8	0x32	VIWU_WCOAL_TMVAL
7:0	0x32	VIWSB_WCOAL_TMVAL

32.1.82 VI_MCCIF_VIWSB_HP_0

Memory Client Interface FIFO Control Register

> **Note:** The FIFO timing aspects of this register are not supported but are retained for software compatibility.

The clock enable fields of this register control the second-level clock gating for the MC side of the MCCIF.

A '1' written to the override field will result in that clock reverting to the legacy mode of operation (where the msc*clk is on whenever the client clk is enabled.

Memory Client High-Priority Control Register

> **Note:** High Priority mode is not supported by the MCCIF clients.
> Registers are retained for backwards compatibility but are not used by the hardware.

Offset: 0x72 | Byte Offset: 0x1c8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	CBW_VIWSB2MC_HPTH

32.1.83 VI_MCCIF_VIWU_HP_0

Memory Client High-Priority Control Register

> **Note:** High Priority mode is not supported by the MCCIF clients.
> Registers are retained for backwards compatibility but are not used by the hardware.

Offset: 0x73 | Byte Offset: 0x1cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	CBW_VIWU2MC_HPTH

32.1.84 VI_MCCIF_VIWV_HP_0

Memory Client High-Priority Control Register

> **Note:** High Priority mode is not supported by the MCCIF clients.
> Registers are retained for backwards compatibility but are not used by the hardware.

Offset: 0x74 | Byte Offset: 0x1d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	CBW_VIWV2MC_HPTH

32.1.85 VI_MCCIF_VIWY_HP_0

Memory Client High-Priority Control Register

> **Note:** High Priority mode is not supported by the MCCIF clients.
> Registers are retained for backwards compatibility but are not used by the hardware.

Offset: 0x75 | Byte Offset: 0x1d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6:0	0x0	CBW_VIWY2MC_HPTH

32.1.86 VI_CSI_PPA_RAISE_FRAME_START_0

CSI Pixel Parser A Raise

CSI Raise vectors

Offset: 0x76 | Byte Offset: 0x1d8 | Read/Write: R/W | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	CSI_PPA_RAISE_FRAME_START_CHANNEL: Raise return channel
15:8	X	CSI_PPA_RAISE_FRAME_START_COUNT: Number of frame starts since the last raise >= count for raise to be returned
4:0	X	CSI_PPA_RAISE_FRAME_START_VECTOR: Raise returned by VI when CSI PPA issues a frame start to the consumer.

32.1.87 VI_CSI_PPA_RAISE_FRAME_END_0

CSI Pixel Parser A Raise

Offset: 0x77 | Byte Offset: 0x1dc | Read/Write: R/W | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	CSI_PPA_RAISE_FRAME_END_CHANNEL: Raise return channel
15:8	X	CSI_PPA_RAISE_FRAME_END_COUNT: Number of frame ends since the last raise >= count for raise to be returned
4:0	X	CSI_PPA_RAISE_FRAME_END_VECTOR: Raise returned by VI when CSI PPA issues a frame end to the consumer.

32.1.88 VI_CSI_PPB_RAISE_FRAME_START_0

CSI Pixel Parser B Raise

Offset: 0x78 | Byte Offset: 0x1e0 | Read/Write: R/W | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	CSI_PPB_RAISE_FRAME_START_CHANNEL: Raise return channel
15:8	X	CSI_PPB_RAISE_FRAME_START_COUNT: Number of frame starts since the last raise >= count for raise to be returned
4:0	X	CSI_PPB_RAISE_FRAME_START_VECTOR: Raise returned by VI when CSI PPB issues a frame start to the consumer.

32.1.89 VI_CSI_PPB_RAISE_FRAME_END_0

CSI Pixel Parser B Raise

Offset: 0x79 | Byte Offset: 0x1e4 | Read/Write: R/W | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	CSI_PPB_RAISE_FRAME_END_CHANNEL: Raise return channel
15:8	X	CSI_PPB_RAISE_FRAME_END_COUNT: Number of frame ends since the last raise >= count for raise to be returned
4:0	X	CSI_PPB_RAISE_FRAME_END_VECTOR: Raise returned by VI when CSI PPB issues a frame end to the consumer.

32.1.90 VI_CSI_PPA_H_ACTIVE_0

VI Horizontal Active

This register defines the horizontal captured (active) area of the input video source with respect to horizontal sync. (This is for CSI data.)

Offset: 0x7a | Byte Offset: 0x1e8 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	CSI_PPA_H_ACTIVE_PERIOD: Horizontal Active Period. This parameter specifies the number of pixels in the horizontal active area. H_ACTIVE_START + H_ACTIVE_PERIOD should be less than 2^NV_VI_H_IN (or 8192). This parameter should be programmed with an even number (bit 16 is ignored internally).
12:0	X	CSI_PPA_H_ACTIVE_START: Horizontal Active Start (offset to active). This parameter specifies the number of clock active edges from horizontal sync active edge to the first horizontal active pixel. If programmed to 0, the first active line starts after the first active clock edge following the horizontal sync active edge.

32.1.91 VI_CSI_PPA_V_ACTIVE_0

Vertical Active

This register defines the vertical captured (active) area of the input video source with respect to vertical sync. (This is for CSI data.)

Offset: 0x7b | Byte Offset: 0x1ec | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	CSI_PPA_V_ACTIVE_PERIOD: Vertical Active Period. This parameter specifies the number of lines in the vertical active area. V_ACTIVE_START + V_ACTIVE_PERIOD should be less than 2^NV_VI_V_IN (or 8192).
12:0	X	CSI_PPA_V_ACTIVE_START: Vertical Active Start (offset to active). This parameter specifies the number of horizontal sync active edges from vertical sync active edge to the first vertical active line. If programmed to 0, the first active line starts after the first horizontal sync active edge following the vertical sync active edge.

32.1.92 VI_CSI_PPB_H_ACTIVE_0

VI Horizontal Active

This register defines the horizontal captured (active) area of the input video source with respect to horizontal sync. (This is for CSI data.)

Offset: 0x7c | Byte Offset: 0x1f0 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	CSI_PPB_H_ACTIVE_PERIOD: Horizontal Active Period. This parameter specifies the number of pixels in the horizontal active area. H_ACTIVE_START + H_ACTIVE_PERIOD should be less than 2^NV_VI_H_IN (or 8192). This parameter should be programmed with an even number (bit 16 is ignored internally).
12:0	X	CSI_PPB_H_ACTIVE_START: Horizontal Active Start (offset to active). This parameter specifies the number of clock active edges from horizontal sync active edge to the first horizontal active pixel. If programmed to 0, the first active line starts after the first active clock edge following the horizontal sync active edge.

32.1.93 VI_CSI_PPB_V_ACTIVE_0

Vertical Active

This register defines the vertical captured (active) area of the input video source with respect to vertical sync. (This is for CSI data.)

Offset: 0x7d | Byte Offset: 0x1f4 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	CSI_PPB_V_ACTIVE_PERIOD: Vertical Active Period. This parameter specifies the number of lines in the vertical active area. V_ACTIVE_START + V_ACTIVE_PERIOD should be less than 2^NV_VI_V_IN (or 8192).
12:0	X	CSI_PPB_V_ACTIVE_START: Vertical Active Start (offset to active). This parameter specifies the number of horizontal sync active edges from vertical sync active edge to the first vertical active line. If programmed to 0, the first active line starts after the first horizontal sync active edge following the vertical sync active edge.

32.1.94 VI_ISP_H_ACTIVE_0

Used when an image comes from ISP

Used only with input from ISP: defines input image horizontal size in pixels

Offset: 0x7e | Byte Offset: 0x1f8 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12:0	X	ISP_H_ACTIVE_PERIOD: Horizontal image size in pixels coming out of the ISP. Must be an even number (bit 0 is ignored).

32.1.95 VI_ISP_V_ACTIVE_0

Used when an image comes from ISP

Used only with input from ISP: defines input image vertical size in lines

Offset: 0x7f | Byte Offset: 0x1fc | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12:0	X	ISP_V_ACTIVE_PERIOD: Vertical image size in lines coming out of the ISP. Must be an even number (bit 0 is ignored).

32.1.96 VI_STREAM_1_RESOURCE_DEFINE_0

This register defines resources used by stream 1.

The field definitions are:

- 0 = resource not used
- 1 = resource used.

Stream raises ("safe to reprogram VI" raises)

The I/O resources used by a data stream going through VI are indicated in STREAM_?_RESOURCE_DEFINE register.

Once resources are set in this register, and after the start of the following picture when ALL the stream's resources are done and idle processing that picture, a raise is generated. It is then safe to reprogram VI's functional units involved in processing that stream.

Two simultaneous data streams are supported, and they do not have to be mutually exclusive.

When no resources are indicated for a stream, no raise is generated.

Offset: 0x80 | Byte Offset: 0x200 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11	0x0	HOST_DMA_BUFFER_OUTPUT_1: 0 = NOT_USED 1 = USED
10	0x0	EPP_OUTPUT_1: 0 = NOT_USED 1 = USED
9	0x0	HOST_DMA_VSYNC_OUTPUT_1: 0 = NOT_USED 1 = USED
8	0x0	SECOND_OUTPUT_1: 0 = NOT_USED 1 = USED
7	0x0	FIRST_OUTPUT_1: 0 = NOT_USED 1 = USED
6	0x0	ISP_INPUT_1: 0 = NOT_USED 1 = USED
5	0x0	CSI_PPB_UNCROPPED_INPUT_1: 0 = NOT_USED 1 = USED
4	0x0	CSI_PPB_CROPPED_INPUT_1: 0 = NOT_USED 1 = USED
3	0x0	CSI_PPA_UNCROPPED_INPUT_1: 0 = NOT_USED 1 = USED
2	0x0	CSI_PPA_CROPPED_INPUT_1: 0 = NOT_USED 1 = USED
1	0x0	HOST_INPUT_1: 0 = NOT_USED 1 = USED
0	0x0	VIP_INPUT_1: Reserved

32.1.97 VI_STREAM_2_RESOURCE_DEFINE_0

Defines resources used by stream 2.

The field definitions are:

- 0 = resource not used
- 1 = resource used.

Offset: 0x81 | Byte Offset: 0x204 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
11	0x0	HOST_DMA_BUFFER_OUTPUT_2: 0 = NOT_USED 1 = USED

Bit	Reset	Description
10	0x0	EPP_OUTPUT_2: 0 = NOT_USED 1 = USED
9	0x0	HOST_DMA_VSYNC_OUTPUT_2: 0 = NOT_USED 1 = USED
8	0x0	SECOND_OUTPUT_2: 0 = NOT_USED 1 = USED
7	0x0	FIRST_OUTPUT_2: 0 = NOT_USED 1 = USED
6	0x0	ISP_INPUT_2: 0 = NOT_USED 1 = USED
5	0x0	CSI_PPB_UNCROPPED_INPUT_2: 0 = NOT_USED 1 = USED
4	0x0	CSI_PPB_CROPPED_INPUT_2: 0 = NOT_USED 1 = USED
3	0x0	CSI_PPA_UNCROPPED_INPUT_2: 0 = NOT_USED 1 = USED
2	0x0	CSI_PPA_CROPPED_INPUT_2: 0 = NOT_USED 1 = USED
1	0x0	HOST_INPUT_2: 0 = NOT_USED 1 = USED
0	0x0	VIP_INPUT_2: Reserved.

32.1.98 VI_RAISE_STREAM_1_DONE_0

Raise vector when all stream 1 resources, as defined by the STREAM_1_RESOURCE_DEFINE register, become idle after the start of the following frame.

Offset: 0x82 | Byte Offset: 0x208 | Read/Write: R/W | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_CHANNEL_STREAM_1
4:0	RW	X	RAISE_VECTOR_STREAM_1

32.1.99 VI_RAISE_STREAM_2_DONE_0

Raise vector when all stream 2 resources, as defined by the STREAM_2_RESOURCE_DEFINE register, become idle after the start of the following frame.

Offset: 0x83 | Byte Offset: 0x20c | Read/Write: R/W | Reset: 0x000X00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_CHANNEL_STREAM_2
4:0	RW	X	RAISE_VECTOR_STREAM_2

32.1.100 VI_TS_MODE_0

ISDB-T Mode Selection Register

ISDB-T tuner mode register settuner/demodulator mode.

Offset: 0x84 | Byte Offset: 0x210 | Read/Write: R/W | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
5:4	X	FLOW_CONTROL_MODE: This field selects the buffer flow control for the Write DMA RDMA: The RDMA engine will release the buffers back to the WDMA as the buffers are consumed NONE: The VI will automatically release the buffer back to the WMDA after each buffer ready is generated. CPU: Software needs to write the TS_CPU_FLOW_CTL register to release each buffer to the WDMA 0 = RDMA 1 = NONE 2 = CPU 3 = RESERVED
3:2	X	PROTOCOL_SELECT: This field selected the pin configuration used for VD[1] NONE: TS_ERROR is tied to 0 TS_PSYNC is tied to 0 TS_ERROR: TS_ERROR is on VD[1] TS_PSYNC is tied to 0 TS_PSYNC: TS_ERROR is tied to 0. TS_PSYNC is on VD[1] 0 = NONE 1 = TS_ERROR 2 = TS_PSYNC 3 = RESERVED
1	X	INPUT_MODE: This field determines if input data is in serial or parallel format 0 = PARALLEL 1 = SERIAL
0	X	ENABLE: This field indicates the global enable for ISDB-T protocol handling 0 = DISABLED 1 = ENABLED

32.1.101 VI_TS_CONTROL_0

ISDB-T Mode Control Register

Offset: 0x85 | Byte Offset: 0x214 | Read/Write: R/W | Reset: 0xXXXX00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1xxx)

Bit	Reset	Description
30:24	X	FEC_SIZE: This field stores the number of FEC bytes to capture (after the BODY has been captured)
23:16	X	BODY_SIZE: This field stores the number of BODY bytes to capture (including PSYNC)
7	X	STORE_UPSTREAM_ERROR_PKTS: This field determines if VI should store packets to memory that have been flagged as UPSTREAM_ERROR packets. DISCARD: Do not store packets in memory STORE: Store UPSTREAM_ERROR packets in memory 0 = DISCARD 1 = STORE
6	X	BODY_VALID_SELECT: This field determines if VALID is used during BODY packet capture IGNORE: the VALID signal is ignored during the capture GATE: the VALID signal gates the capture of BODY data. 0 = IGNORE 1 = GATE

Bit	Reset	Description
5:4	X	START_SELECT: This field defines how the START of packet condition is determined PSYNC: PSYNC assertion rising edge VALID: VALID assertion rising edge BOTH: PSYNC && VALID asserted rising edge 0 = RESERVED 1 = PSYNC 2 = VALID 3 = BOTH
3	LOW	CLK_POLARITY: 0 = HIGH 1 = LOW
2	X	ERROR_POLARITY: 0 = HIGH 1 = LOW
1	X	PSYNC_POLARITY: 0 = HIGH 1 = LOW
0	X	VALID_POLARITY: This field indicates the polarity of TS_VALID. Only has affect when TS_MODE.ENABLE == ENABLED LOW indicates that the polarity of TS_VALID is active low. HIGH indicates that the polarity of TS_VALID is active high. 0 = HIGH 1 = LOW

32.1.102 VI_TS_PACKET_COUNT_0

ISDB-T Packet Count Register

Offset: 0x86 | Byte Offset: 0x218 | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
16	X	TS_PACKET_COUNT_VALUE_OVERFLOW: This field is set to OVERFLOW when the value passes from 0xFFFF to 0x0000. It stays high until the CPU writes a zero to this bit to reset it. 0 = NONE 1 = OVERFLOW
15:0	X	TS_PACKET_COUNT_VALUE: This field holds the current value of the received packet counter. This counter increments in the presence of a new packet, regardless of whether it is flagged as an error. The counter can be cleared by writing this register with 0's and can also be preloaded to any value by writing the preload value to the register.

32.1.103 VI_TS_ERROR_COUNT_0

ISDB-T Error Count Register

Offset: 0x87 | Byte Offset: 0x21c | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
16	X	TS_ERROR_COUNT_VALUE_OVERFLOW: This field is set to OVERFLOW when the value passes from 0xFFFF to 0x0000. It stays high until the CPU writes a zero to this bit to reset it. 0 = NONE 1 = OVERFLOW
15:0	X	TS_ERROR_COUNT_VALUE: This field holds the current value of the error packet counter. This counter increments in the presence of a packet flagged as error (see TS_ERROR) 0000 or a detected protocol violation. The counter can be cleared by writing this register with 0's and can also be preloaded to any value by writing the preload value to the register.

32.1.104 VI_TS_CPU_FLOW_CTL_0

ISDB-T CPU Flow Control Register

Offset: 0x88 | Byte Offset: 0x220 | Read/Write: RO | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	BUFFER_RELEASE: Used only when the FLOW_CONTROL_MODE register is set to CPU. Software must write this register to release each buffer back to WDMA. Failure to write this register when buffers are consumed will result in the WDMA stalling when it consumes all allocated/free buffers.

32.1.105 VI_VB0_CHROMA_BUFFER_STRIDE_FIRST_0

Video Buffer Set 0 Chroma Buffer Stride

This feature represents an alternative value to using VB0_BUFFER_STRIDE_C.

HOST_DMA_WRITE_BUFFER.BUFFER_SIZE (bytes) is used to hold the number of bytes in a buffer for ISDB-T mode.

Offset: 0x89 | Byte Offset: 0x224 | Read/Write: R/W | Reset: 0xXXXXXXXX (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	VB0_CHROMA_BUFFER_STRIDE_SELECT: select type of Chroma buffer stride: 0 = Use VB0_BUFFER_STRIDE_C, deriving chroma buffer stride from luma buffer stride (default). 1 = Use VB0_CHROMA_BUFFER_STRIDE. 0 = RATIO 1 = VALUE
29:0	X	VB0_CHROMA_BUFFER_STRIDE: Chroma buffer stride in bytes

32.1.106 VI_VB0_CHROMA_LINE_STRIDE_FIRST_0

Video Buffer Set 0 chroma line stride for First Output of planar YUV formats

Offset: 0x8a | Byte Offset: 0x228 | Read/Write: R/W | Reset: 0x0000XXXX (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	VB0_CHROMA_LINE_STRIDE_SELECT: select type of Chroma line stride: 0 = Use VB0_H_SIZE_1, deriving chroma line stride from luma line stride (default). 1 = Use VB0_CHROMA_H_SIZE_1. 0 = RATIO 1 = VALUE
12:0	X	VB0_CHROMA_H_SIZE_1: Video Buffer Set 0 chroma horizontal size This parameter specifies the chroma line stride (in pixels) for lines in the video buffer set 0. This parameter must be programmed as a multiple of 4 pixels (bits 1-0 are ignored).

32.1.107 VI_EPP_LINES_PER_BUFFER_0

Number of buffers per output frame in EPP

This register is used for VI2EPP syncpt only.

The VI will based on num_lines = frame_height/EPP_NUM_OF_BUFFER_PER_FRAME, send vi2epp_trigger for every num_lines

Offset: 0x8b | Byte Offset: 0x22c | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12:0	X	LINES_PER_BUFFER: maximum 256 buffers per frame. linesPerBuffer = FLOOR(eppLineCount/eppBufferCount) linesPerBuffer must be > 2 eppLineCount must take into account any cropping in the EPP.

32.1.108 VI_BUFFER_RELEASE_OUTPUT1_0

Writes to this register will decrease BUFFER_COUNTER by 1

Offset: 0x8c | Byte Offset: 0x230 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	BUFFER_RELEASE_OUTPUT1

32.1.109 VI_BUFFER_RELEASE_OUTPUT2_0

Offset: 0x8d | Byte Offset: 0x234 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	BUFFER_RELEASE_OUTPUT2

32.1.110 VI_DEBUG_FLOW_CONTROL_COUNTER_OUTPUT1_0

Debug register

Offset: 0x8e | Byte Offset: 0x238 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	BUFFER_COUNT_OUTPUT1

32.1.111 VI_DEBUG_FLOW_CONTROL_COUNTER_OUTPUT2_0

Offset: 0x8f | Byte Offset: 0x23c | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7:0	X	BUFFER_COUNT_OUTPUT2

32.1.112 VI_TERMINATE_BW_FIRST_0

Writes to this register will terminate MC on BW operation in the FIRST output.

Offset: 0x90 | Byte Offset: 0x240 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TERMINATE_FIRST_BW

32.1.113 VI_TERMINATE_BW_SECOND_0

Writes to this register will terminate MC on BW operation in the SECOND output.

Offset: 0x91 | Byte Offset: 0x244 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	TERMINATE_SECOND_BW

32.1.114 VI_VB0_FIRST_BUFFER_ADDR_MODE_0

Memory Controller Tiling Definitions

Offset: 0x92 | Byte Offset: 0x248 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0)

Bit	Reset	Description
8	0x0	UV1_TILE_MODE: 0 = LINEAR 1 = TILED
0	0x0	Y1_TILE_MODE: 0 = LINEAR 1 = TILED

32.1.115 VI_VB0_SECOND_BUFFER_ADDR_MODE_0

Offset: 0x93 | Byte Offset: 0x24c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	Y2_TILE_MODE: 0 = LINEAR 1 = TILED

32.1.116 VI_RESERVE_0_0

Reserved register

Offset: 0x94 | Byte Offset: 0x250 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:12	X	nc_RESERVE_0_3
11:8	X	nc_RESERVE_0_2
7:4	X	nc_RESERVE_0_1
3:0	X	nc_RESERVE_0_0

32.1.117 VI_RESERVE_1_0

Reserved register

Offset: 0x95 | Byte Offset: 0x254 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:12	X	nc_RESERVE_1_3
11:8	X	nc_RESERVE_1_2
7:4	X	nc_RESERVE_1_1
3:0	X	nc_RESERVE_1_0

32.1.118 VI_RESERVE_2_0

Reserved register

Offset: 0x96 | Byte Offset: 0x258 | Read/Write: R/W | Reset: 0x0000XXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
15:12	X	nc_RESERVE_2_3

Bit	Reset	Description
11:8	X	nc_RESERVE_2_2
7:4	X	nc_RESERVE_2_1
3:0	0x0	nc_RESERVE_2_0

32.1.119 VI_RESERVE_3_0

Reserved register

Offset: 0x97 | Byte Offset: 0x25c | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:12	X	nc_RESERVE_3_3
11:8	X	nc_RESERVE_3_2
7:4	X	nc_RESERVE_3_1
3:0	X	nc_RESERVE_3_0

32.1.120 VI_RESERVE_4_0

Reserved register

Offset: 0x98 | Read/Write: R/W | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:12	X	nc_RESERVE_4_3
11:8	X	nc_RESERVE_4_2
7:4	X	nc_RESERVE_4_1
3:0	X	nc_RESERVE_4_0

32.1.121 VI_MCCIF_VIWSB_HYST_0

Memory Client Interface FIFO Control Register

> **Note:** The FIFO timing aspects of this register are not supported but are retained for software compatibility.

The clock enable fields of this register control the second-level clock gating for the MC side of the MCCIF.

A '1' written to the override field will result in that clock reverting to the legacy mode of operation (where the msc*clk is on whenever the client clock is enabled.

Memory Client Hysteresis Control Register

> **Note:** Hysteresis is not supported by the MCCIF clients.
> Registers are retained for backwards compatibility but are not used by the hardware.

Offset: 0x9a | Byte Offset: 0x268 | Read/Write: R/W | Reset: 0xc00001ff (0b1100xxxxxxxxxxxxxxx000111111111)

Bit	Reset	Description
31	ENABLE	CBW_VIWSB2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBW_VIWSB2MC_HYST_REQ_TH

Bit	Reset	Description
11:0	0x1ff	CBW_VIWSB2MC_HYST_REQ_TM

32.1.122 VI_MCCIF_VIWU_HYST_0

Memory Client Hysteresis Control Register

> **Note:** Hysteresis is not supported by the MCCIF clients.
> Registers are retained for backwards compatibility but are not used by the hardware.

Offset: 0x9b | Byte Offset: 0x26c | Read/Write: R/W | Reset: 0xc00001ff (0b1100xxxxxxxxxxxxxxxx000111111111)

Bit	Reset	Description
31	ENABLE	CBW_VIWU2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBW_VIWU2MC_HYST_REQ_TH
11:0	0x1ff	CBW_VIWU2MC_HYST_REQ_TM

32.1.123 VI_MCCIF_VIWV_HYST_0

Memory Client Hysteresis Control Register

> **Note:** Hysteresis is not supported by the MCCIF clients.
> Registers are retained for backwards compatibility but are not used by the hardware.

Offset: 0x9c | Byte Offset: 0x270 | Read/Write: R/W | Reset: 0xc00001ff (0b1100xxxxxxxxxxxxxxxx000111111111)

Bit	Reset	Description
31	ENABLE	CBW_VIWV2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBW_VIWV2MC_HYST_REQ_TH
11:0	0x1ff	CBW_VIWV2MC_HYST_REQ_TM

32.1.124 VI_MCCIF_VIWY_HYST_0

Memory Client Hysteresis Control Register

> **Note:** Hysteresis is not supported by the MCCIF clients.
> Registers are retained for backwards compatibility but are not used by the hardware.

Offset: 0x9d | Byte Offset: 0x274 | Read/Write: R/W | Reset: 0xc00001ff (0b1100xxxxxxxxxxxxxxxx000111111111)

Bit	Reset	Description
31	ENABLE	CBW_VIWY2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBW_VIWY2MC_HYST_REQ_TH
11:0	0x1ff	CBW_VIWY2MC_HYST_REQ_TM

32.1.125 VI_FIELD_STATUS_SHADOW1_0

This register contains a copy of the field status and frame counter registers at the end of the previous frame. The register is updated when the OP_DONE syncpt fires at the end of the frame. So the value should be "safe" to read for the entire "next" frame. There are 2 registers, each of which is tied to one of the 2 output channels

Offset: 0x9e | Byte Offset: 0x278 | Read/Write: RO | Reset: 0xXXXX000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	FRAME_COUNT_SHADOW1: Shadow copy of the number of frames received (VSYNCs)
0	X	FIELD_STATUS_SHADOW1: Shadow copy of the field count register - synced to channel 1

32.1.126 VI_FIELD_STATUS_SHADOW2_0

Offset: 0x9f | Byte Offset: 0x27c | Read/Write: RO | Reset: 0xXXXX000X (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	FRAME_COUNT_SHADOW2: Shadow copy of the number of frames received (VSYNCs)
0	X	FIELD_STATUS_SHADOW2: Shadow copy of the field count register - synced to channel 2

32.1.127 VI_COREMUX_CROP_H_ACTIVE_0

VI Horizontal Active

This register defines the horizontal captured (active) area of the input video source with respect to horizontal sync.

Offset: 0xa0 | Byte Offset: 0x280 | Read/Write: R/W | Reset: 0xXXXXXXXX (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	COREMUX_CROP_ENABLE: enable cropping after core-mux 0 = DISABLED 1 = ENABLED
28:16	X	COREMUX_CROP_H_ACTIVE_PERIOD: Horizontal Active Period. This parameter specifies the number of pixels in the horizontal active area. H_ACTIVE_START + H_ACTIVE_PERIOD should be less than 2^NV_VI_H_IN (or 8192). This parameter should be programmed with an even number (bit 16 is ignored internally).
12:0	X	COREMUX_CROP_H_ACTIVE_START: Horizontal Active Start (offset to active). This parameter specifies the number of clock active edges from horizontal sync active edge to the first horizontal active pixel. If programmed to 0, the first active line starts after the first active clock edge following the horizontal sync active edge.

32.1.128 VI_COREMUX_CROP_V_ACTIVE_0

Vertical Active

This register defines the vertical captured (active) area of the input video source with respect to vertical sync.

Offset: 0xa1 | Byte Offset: 0x284 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	COREMUX_CROP_V_ACTIVE_PERIOD: Vertical Active Period. This parameter specifies the number of lines in the vertical active area. V_ACTIVE_START + V_ACTIVE_PERIOD should be less than 2^NV_VI_V_IN (or 8192).
12:0	X	COREMUX_CROP_V_ACTIVE_START: Vertical Active Start (offset to active). This parameter specifies the number of horizontal sync active edges from vertical sync active edge to the first vertical active line. If programmed to 0, the first active line starts after the first horizontal sync active edge following the vertical sync active edge.

32.1.129 VI_ALTMUX_CROP_H_ACTIVE_0

VI Horizontal Active

This register defines the horizontal captured (active) area of the input video source with respect to horizontal sync.

Offset: 0xa2 | Byte Offset: 0x288 | Read/Write: R/W | Reset: 0xXXXXXXXX (0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	ALTMUX_CROP_ENABLE: enable cropping after core-mux 0 = DISABLED 1 = ENABLED
28:16	X	ALTMUX_CROP_H_ACTIVE_PERIOD: Horizontal Active Period. This parameter specifies the number of pixels in the horizontal active area. H_ACTIVE_START + H_ACTIVE_PERIOD should be less than 2^NV_VI_H_IN (or 8192). This parameter should be programmed with an even number (bit 16 is ignored internally).
12:0	X	ALTMUX_CROP_H_ACTIVE_START: Horizontal Active Start (offset to active). This parameter specifies the number of clock active edges from horizontal sync active edge to the first horizontal active pixel. If programmed to 0, the first active line starts after the first active clock edge following the horizontal sync active edge.

32.1.130 VI_ALTMUX_CROP_V_ACTIVE_0

Vertical Active

This register defines the vertical captured (active) area of the input video source with respect to vertical sync.

Offset: 0xa3 | Byte Offset: 0x28c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	ALTMUX_CROP_V_ACTIVE_PERIOD: Vertical Active Period. This parameter specifies the number of lines in the vertical active area. V_ACTIVE_START + V_ACTIVE_PERIOD should be less than 2^NV_VI_V_IN (or 8192).
12:0	X	ALTMUX_CROP_V_ACTIVE_START: Vertical Active Start (offset to active). This parameter specifies the number of horizontal sync active edges from vertical sync active edge to the first vertical active line. If programmed to 0, the first active line starts after the first horizontal sync active edge following the vertical sync active edge.

32.1.131 VI_VI_STEREO_CONTROL_0

This register controls the aspects of stereo capture.

Offset: 0xa4 | Byte Offset: 0x290 | Read/Write: R/W | Reset: 0xXXXX0XX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
31:24	X	HBLANK_WIDTH: number of blank pixels inserted at the end of the super line.
21:16	X	SEAM_WIDTH: number of blank pixels inserted between lines
8	X	STEREO_BUFFER_1_SRC: source for stereo buffer 1 0 = CSIA 1 = CSIB
4	X	STEREO_BUFFER_0_SRC: source for stereo buffer 0 0 = CSIA 1 = CSIB
0	DISABLED	ENABLE_STEREO: enable stereo capture. Lines from two sources will be concatenated and sent to ISP 0 = DISABLED 1 = ENABLED

32.1.132 VI_VI_CSIA_PATTERN_GEN_SIZE_0

Control register for CSI-A Pattern Generator

Offset: 0xa5 | Byte Offset: 0x294 | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:16	X	CSIA_PG_V_COUNT_MAX: Maximum Y of pattern (program to 1 less than pattern height)
13:0	X	CSIA_PG_H_COUNT_MAX: Maximum X of pattern (program to 1 less than pattern width)

32.1.133 VI_VI_CSIA_PATTERN_GEN_CONTROL_0

Offset: 0xa6 | Byte Offset: 0x298 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxx00xxxx0010)

Bit	Reset	Description
9:8	0x0	CSIA_PG_UPDATE_CYCLES: "dead" cycles between updates of pattern generator, i.e., 0 = update every cycle, 1 = update every other cycle
3	0x0	CSIA_PG_ZERO_V_COUNT: Hold V count to zero when generating pattern data
2	0x0	CSIA_PG_ZERO_H_COUNT: Hold H count to zero when generating pattern data
1	BAYER	CSIA_PG_BAYER: Enable pattern generator 0 = YUV 1 = BAYER
0	DISABLED	CSIA_PG_ENABLE: enable pattern generator 0 = DISABLED 1 = ENABLED

32.1.134 VI_VI_CSIB_PATTERN_GEN_SIZE_0

Control Register for CSI-B pattern generator

Offset: 0xa7 | Byte Offset: 0x29c | Read/Write: R/W | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
29:16	X	CSIB_PG_V_COUNT_MAX: Maximum Y of pattern (program to 1 less than pattern height)
13:0	X	CSIB_PG_H_COUNT_MAX: Maximum X of pattern (program to 1 less than pattern width)

32.1.135 VI_VI_CSIB_PATTERN_GEN_CONTROL_0

Offset: 0xa8 | Byte Offset: 0x2a0 | Read/Write: R/W | Reset: 0x00000002 (0bxxxxxxxxxxxxxxxxxxxxxx00xxxx0010)

Bit	Reset	Description
9:8	0x0	CSIB_PG_UPDATE_CYCLES: "dead" cycles between updates of pattern generator, i.e., 0 = update every cycle, 1 = update every other cycle
3	0x0	CSIB_PG_ZERO_V_COUNT: hold V count to zero when generating pattern data
2	0x0	CSIB_PG_ZERO_H_COUNT: hold H count to zero when generating pattern data
1	BAYER	CSIB_PG_BAYER: enable pattern generator 0 = YUV 1 = BAYER
0	DISABLED	CSIB_PG_ENABLE: enable pattern generator 0 = DISABLED 1 = ENABLED

32.2 MIPI-CSI Registers

Refer to the CSI section in this document for descriptions of these registers.

33.0 SD/MMC CONTROLLER

The SD/MMC Controller can interface with SD, SDIO, or eMMC devices. It supports both on-board devices and plug-in cards.

Each Tegra® 4 device contains four instances of this controller, each of which is identical in its internal function, but they vary in I/O capabilities:

Controller	UHS-I Signaling Supported	8-bit eMMC Supported	Signaling Voltages
SDMMC1	Yes	No	1.8, 2.8-3.3 V
SDMMC2	Yes	Yes	1.8, 2.8-3.3 V
SDMMC3	Yes	No	1.8, 2.8-3.3 V
SDMMC4	Yes	Yes	1.2, 1.8 V

33.1 Standards Supported

The SD/MMC Controller supports the following industry standards at the stated levels.

- "SD Specifications Part E1 SDIO Specification", Technical Committee SD Card Association, Version 3.00, 16-December 2010
- "SD Specifications Part 1 Physical Layer Specification", Technical Committee SD Card Association, Version 3.01, 18-February 2010.
- "SD Specifications Part A2 SD Host Controller Standard Specification", Technical Committee SD Association, Version 3.00, 18-February 2010.
- JEDEC eMMC Electrical Interface Specification, Version 4.51, JEDEC Solid State Technology Association.

33.2 Speeds Supported

These tables show the maximum data rates available over the physical interface. The speeds achievable in actual use will be less than these rates, depending on the performance of the device itself, protocol limitations, and the software driver.

For SD 3.0 data transfer modes, these are the maximum data transfer speeds:

Speed Mode	(removable) Signal Voltage	I/O Frequency (MHz)	Bus Width	Maximum Throughput (MBytes/s)
Default speed	3.3	25	1,4	12.5
High Speed	3.3	50	1,4	25
SDR12	1.8	25	1,4	12.5
SDR25	1.8	50	1,4	25
SDR50	1.8	100	1,4	50
SDR104 UHS-I	1.8	208	1,4	104

For eMMC 4.51 data transfer modes, these are the maximum data transfer speeds:

Speed Mode	I/O Frequency (MHz)	Bus Width	Maximum Throughput (MBytes/s)
Legacy Speed	26	1,4,8	26
High Speed SDR	52	1,4,8	52
High Speed DDR	52	4,8	104
HS200 (SDR)	200	4,8	200

33.3 Operation

33.3.1 Hardware / Software Partitioning

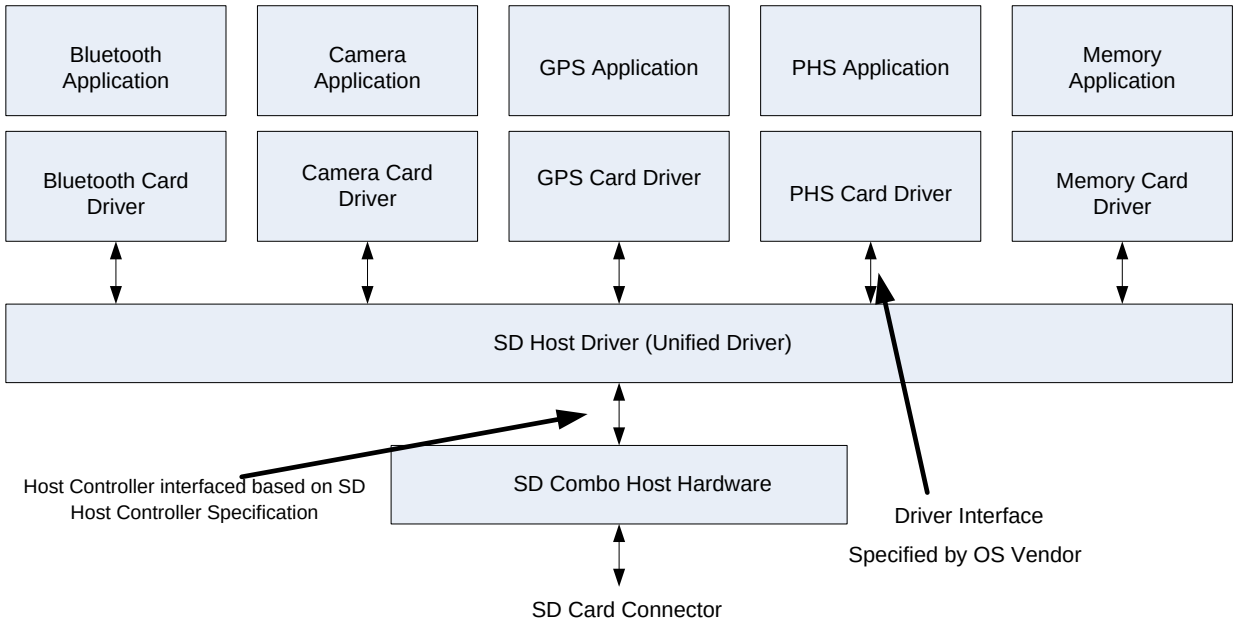
Hardware

The role of the hardware controller is to send out the programmed command, store the response, and make it visible to software. It updates the status registers and generates interrupts when attention is required. Software may also configure it for DMA operation.

Software

The software driver determines which type of card is inserted in the slot by sending the initialization commands and observing the responses received from the card. After identifying the card that is inserted, it should only program the corresponding set of commands that are applicable. It can enable interrupts for which notification is desired.

Figure 126: Host Hardware and Driver Architecture



33.4 Caveats and Assumptions

- A single SD/MMC controller handles only one device at a time.
- The software should configure SDMMC_VENDOR_CLOCK_CNTRL_0_SDMMC_CLK to _DISABLE before turning off the SD/MMC clock (and configure back to _ENABLE after turning on the SD/MMC clock). This is required in order to support asynchronous card interrupts when no clock is supplied to the card.
- PMC can be configured to wake the Tegra 4 device from the LP0 power state based on either:
  - Card Detection pin.
  - Asynchronous interrupt on the DAT1 line
- Write protect and Card detect logic is implemented in Tegra for SDMMC1 and SDMMC3 only.
- Off-card ECC, described in the MMC specification, is not supported.

33.5 Programming Guidelines

This section assumes a SD Host driver complying with the SD Specifications' Part A2, SD Host Control Standard Specification. The following subsections detail the necessary changes required for fully featured SD (and eMMC boot mode) operation.

33.5.1 Initialization

The following register writes should be done before the SD host driver is loaded. These register writes can be done in any order, and writes to different register fields within the same register should ideally be combined into a single write for efficiency:

33.5.1.1 General

These settings apply to all SDMMC controllers in use:

- Enable tuning in SDR50 by setting SDMMC_VENDOR_CLOCK_CNTRL_0_SDR50_TUNING_OVERRIDE.
- Enable SDR104 (HS200 for eMMC application) support by setting UHS_MODE_SEL to SDR104 in HOST_CONTROL_2 (offset 03Eh) register.
- Clamp clocks during resets by setting these two values:
  - SDMMC_VENDOR_CLOCK_CNTRL_0_PADPIPE_CLKEN_OVERRIDE_NORMAL
  - SDMMC_VENDOR_CLOCK_CNTRL_0_SPI_MODE_CLKEN_OVERRIDE_NORMAL

33.5.1.2 SDMMC1

These settings are specific to the SDMMC1 controller:

Supply Voltage	33 Ohm		50 Ohm		66 Ohm		100 Ohm	
	UP	DN	UP	DN	UP	DN	UP	DN
1.8 V	6'h46	6'h36	6'h2A	6'h20	6'h22	6'h17	6'h13	6'hB
3.3 V	-		6'h24	6'h14	-		-	

UP - APB_MISC_GP_SDIO1CFGPADCTRL_0_CFG2TMC_SDIO1CFG_CAL_DRVUP

DN - APB_MISC_GP_SDIO1CFGPADCTRL_0_CFG2TMC_SDIO1CFG_CAL_DRVDN

- APB_MISC_GP_SDIO1CFGPADCTRL_0_CFG2TMC_SDIO1CFG_CAL_DRVDN_SLWR **= 2'h0**
- APB_MISC_GP_SDIO1CFGPADCTRL_0_CFG2TMC_SDIO1CFG_CAL_DRVUP_SLWF **= 2'h0**
  SDMMC_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL **= 4'h7**
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = **8'h0**
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = **8'h0**

- If using the SDMMC1 interface, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_SDMMC1_CLK_0_PULL_UP_NORMAL
  - If using eMMC (a stronger external 4.7 kOhm pull-up is used on the CMD line):
    - PINMUX_AUX_SDMMC1_CMD_0_PULL_UP_NORMAL
  - If using SD or SDIO:
    - PINMUX_AUX_SDMMC1_CMD_0_PULL_UP_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT3_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT2_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT1_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT0_0_PUPD_PULL_UP

33.5.1.3 SDMMC2

These settings are specific to the SDMMC2 controller:

Supply Voltage	33 Ohm		50 Ohm		66 Ohm		100 Ohm	
	UP	DN	UP	DN	UP	DN	UP	DN
1.8 V	6'h46	6'h36	6'h2A	6'h20	6'h22	6'h17	6'h13	6'hB
3.3 V	-		6'h24	6'h14	-		-	

UP - APB_MISC_GP_ATCFG6PADCTRL_0_CFG2TMC_ATCFG6_CAL_DRVUP

DN - APB_MISC_GP_ATCFG6PADCTRL_0_CFG2TMC_ATCFG6_CAL_DRVDN

- APB_MISC_GP_ATCFG6PADCTRL_0_CFG2TMC_SDIO2CFG_CAL_DRVDN_SLWR **= 2'h0**
- APB_MISC_GP_ATCFG6PADCTRL_0_CFG2TMC_SDIO2CFG_CAL_DRVUP_SLWF **= 2'h0**
- SDMMC_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL **= 4'h7**
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = **8'h0**
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = **8'h0**

- If using the SDMMC2 controller's external interface shared with the GMI pins, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_ GMI_CLK_0 _PUPD_PUUL_DOWN
  - If using eMMC (a stronger external 4.7 kOhm pull-up is used on the CMD line):

- PINMUX_AUX_GMI_AD15_0_PUPD_NORMAL

If using SD or SDIO:

- PINMUX_AUX_GMI_AD15_0_PUPD_PULL_UP

- PINMUX_AUX_GMI_IORDY_0_PUPD_PULL_UP
- PINMUX_AUX_GMI_AD12_0_PUPD_PULL_UP
- PINMUX_AUX_GMI_AD13_0_PUPD_PULL_DOWN
- PINMUX_AUX_GMI_AD14_0_PUPD_PULL_UP
- PINMUX_AUX_GMI_CS2_N_0_PUPD_PULL_UP
- PINMUX_AUX_GMI_CS3_N_0_PUPD_PULL_UP
- PINMUX_AUX_GMI_DQS_P_0_PUPD_PULL_UP
- PINMUX_AUX_GMI_CS7_N_0_PUPD_PULL_UP

33.5.1.4 SDMMC3

These settings are specific to the SDMMC3 controller:

Supply Voltage	33 Ohm		50 Ohm		66 Ohm		100 Ohm	
	UP	DN	UP	DN	UP	DN	UP	DN
1.8 V	6'h46	6'h36	6'h2A	6'h20	6'h22	6'h17	6'h13	6'hB
3.3 V	-		6'h24	6'h14	-		-	

UP - APB_MISC_GP_SDIO3CFGPADCTRL_0_CFG2TMC_SDIO3CFG_CAL_DRVUP

DN - APB_MISC_GP_SDIO3CFGPADCTRL_0_CFG2TMC_SDIO3CFG_CAL_DRVDN

- APB_MISC_GP_SDIO3CFGPADCTRL_0_CFG2TMC_SDIO3CFG_CAL_DRVDN_SLWR = **2'h0**
- APB_MISC_GP_SDIO3CFGPADCTRL_0_CFG2TMC_SDIO3CFG_CAL_DRVUP_SLWF = **2'h0**
- SDMMC_SDMEMCOMPPADCTRL_2_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL = **4'h7**
  SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = **8'h0**
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = **8'h0**

- If using the SDMMC3 interface, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_SDMMC3_CLK_0_PUPD_NORMAL
  - If using eMMC (a stronger external 4.7 kOhm pullup is used on the CMD line):
    - PINMUX_AUX_SDMMC3_CMD_0_PUPD_NORMAL
  - If using SD or SDIO:
    - PINMUX_AUX_SDMMC3_CMD_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT3_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT2_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT1_0_PUPD_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT0_0_PUPD_PULL_UP

33.5.1.5 SDMMC4

These settings are specific to the SDMMC4 controller:

Supply Voltage	33 Ohms		50 Ohms	
	UP	DN	UP	DN
1.8 V	6'h07	6'h07	6'h02	6'h01
1.2 V	6'h0E	6'h0E	6'h05	6'h05

UP - APB_MISC_GP_GMACFGPADCTRL_0_CFG2TMC_GMACFG_CAL_DRVUP

DN - APB_MISC_GP_GMACFGPADCTRL_0_CFG2TMC_GMACFG_CAL_DRVDN

- APB_MISC_GP_GMACFGPADCTRL_0_CFG2TMC_GMACFG_CAL_DRVDN_SLWR **= 2'h0**
- APB_MISC_GP_GMACFGPADCTRL_0_CFG2TMC_GMACFG_CAL_DRVUP_SLWF **= 2'h0**
- SDMMC_SDMEMCOMPPADCTRL_0_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL **= 4'h7**
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_ENABLE_ENABLED
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PD_OFFSET = 8'h0
- SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8'h0
-
- If using the SDMMC4 interface, correctly enable the integrated pullup/down resistors there:
- PINMUX_AUX_SDMMC4_CLK_0_PUPD_PULL_DOWN
- If using eMMC (a stronger external 4.7 kOhm pullup is used on the CMD line):
  - PINMUX_AUX_SDMMC4_CMD_0_PUPD_NORMAL
- If using SD or SDIO:
  - PINMUX_AUX_SDMMC4_CMD_0_PUPD_PULL_UP
- PINMUX_AUX_SDMMC4_DAT7_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT6_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT5_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT4_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT3_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT2_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT1_0_PULL_UP_PULL_UP
- PINMUX_AUX_SDMMC4_DAT0_0_PULL_UP_PULL_UP

33.5.2 eMMC Boot Modes

The section of this document on the boot process describes the boot process in more detail, but note that only the SDMMC4 controller can be used as the e.MMC boot device.

33.5.3 Boot Option1

1. The clock to the card needs to be configured to 26 MHz.
2. Program the number of blocks. Configuring block length has no effect; it is always fixed to 512 bytes.
3. Configuring the data transfer direction has no impact, since it taken as CARD to HOST for boot mode.
4. Select either SDMA or PIO mode. (Tegra 4 devices do not support ADMA modes during boot option 1.)

5. Configure **SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0** – The maximum timeout value that needs to be programmed is 50ms.The value programmed should exclude the 74 cycles that are required to enter boot mode.
6. Configure **SDMMC_VENDOR_BOOT_DAT_TIMEOUT_0** – The max timeout value that needs to be programmed is 1 second .The value programmed should exclude the 74cycles that are required to enter boot mode.
7. Configure **SDMMC_VENDOR_BOOT_CNTRL_0**[1] to 1 to ask the engine to look for boot_ack.
8. Configure **SDMMC_VENDOR_BOOT_CNTRL_0**[0] to trigger the engine.
9. If **SDMMC_INTERRUPT_STATUS_0**[31] = 1,the BOOT_ACK is not equal to 00101.The engine just informs the software and continues to fetch data from the card
10. If **SDMMC_INTERRUPT_STATUS_0**[30] = 1, the BOOT_ACK timeout has occurred. The engine just informs the software and continues to fetch data from the card.
11. The software should look for transfer complete interrupt. If “data_time_out” error has occurred, then the data transfer is not successful.
12. After successful data transfer from card, the software should look for “sdma_engine_busy” bit of **SDMMC_OBS_SDMMC_DBG0_0 [0].**Software shouldn’t program commands until this bit is zero.

33.5.4 Boot Option2

Boot option2 is treated like any normal read command. The BOOT_ACK should be enabled if the card supports boot acknowledgment. **SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0, SDMMC_VENDOR_BOOT_DAT_TIMEOUT_0** should be programmed based on the frequency of operation.

33.5.5 Card Detect and Write Protect

Only SDMMC1 and SDMMC3 support removable SD cards. The standard SD Host, SDCD# pin acts as card detect pin and CDWP# pin acts as Write protect pin. The SD Card State registers can be ignored for embedded on-board devices.

PIN MUX detail for CD and WP pins for SDMMC1 and SDMMC3.

PINMUX_AUX_SDMMC1_WP_N_0_PUPD_PULL_UP

PINMUX_AUX_KB_COL4_0_PUPD_PULL_UP - This is the Pinmux option for SDMMC1 CD

PINMUX_AUX_KB_COL4_0_PUPD_PULL_UP - This is the Pinmux option for SDMMC3 WP

PINMUX_AUX_SDMMC3_CD_N_0_PUPD_PULL_UP

Additionally, an abort command should be issued if a card is removed during an SD transfer.

33.5.6 Improving Non-DMA (PIO) performance

When transferring more than a single 512 byte block size, PIO performance is reduced because the standard process waits for the buffer ready interrupt before beginning to move the next block of data.

Next block data movement can instead be pipelined on a four-byte word basis. To do this, software should waiting for SDMMC_PRESENT_STATE_0’s BUFFER_READ_EN or BUFFER_WRITE_EN fields to assert instead of the buffer ready interrupt. These status fields signify that the next four-byte word can be read from or written to the internal 512 byte buffer.

33.5.7 SD3.0 Signal Voltage Switching

Note:	This section is applicable only to the UHS-I SDMMC1, SDMMC2 and SDMMC3 controllers. The SDMMC4 controllers do not require these settings.

Signal Voltage Switching happens after setting 1.8V Signal enable in the host control 2 register. Switching the voltage from 3.3V to 1.8V, UHS-I signaling requires switching the external power supply.

After setting 1.8V Signal Enable in the *Host Control 2* register, the SD host driver communicates with the hardware platform to change the SDMMC1, SDMMC2 and SDMMC3 I/O voltage from 3.3 to 1.8V. This is typically via I2C control of an external "PMIC" (Power Management IC).

When resetting the controller (e.g., when an SD3.0 card in 1.8V mode has been removed), the SD host driver again communicates with the system software/platform to increase the SD I/O voltage back to 3.3V.

33.5.7.1 Initialization

Before the standard SD driver begins running:

1. Route the SDMMC controller's system interrupt to the appropriate driver.
2. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_VOLT_SWITCH_INT_STATUS to clear any latent interrupts.
3. Set SDMMC_VENDOR_SYS_SW_CNTRL_0_VOLT_SWITCH_INT _ENABLE to enable the system interrupt on an SD3.0 UHS-I Voltage Switch event.

33.5.7.2 Entry

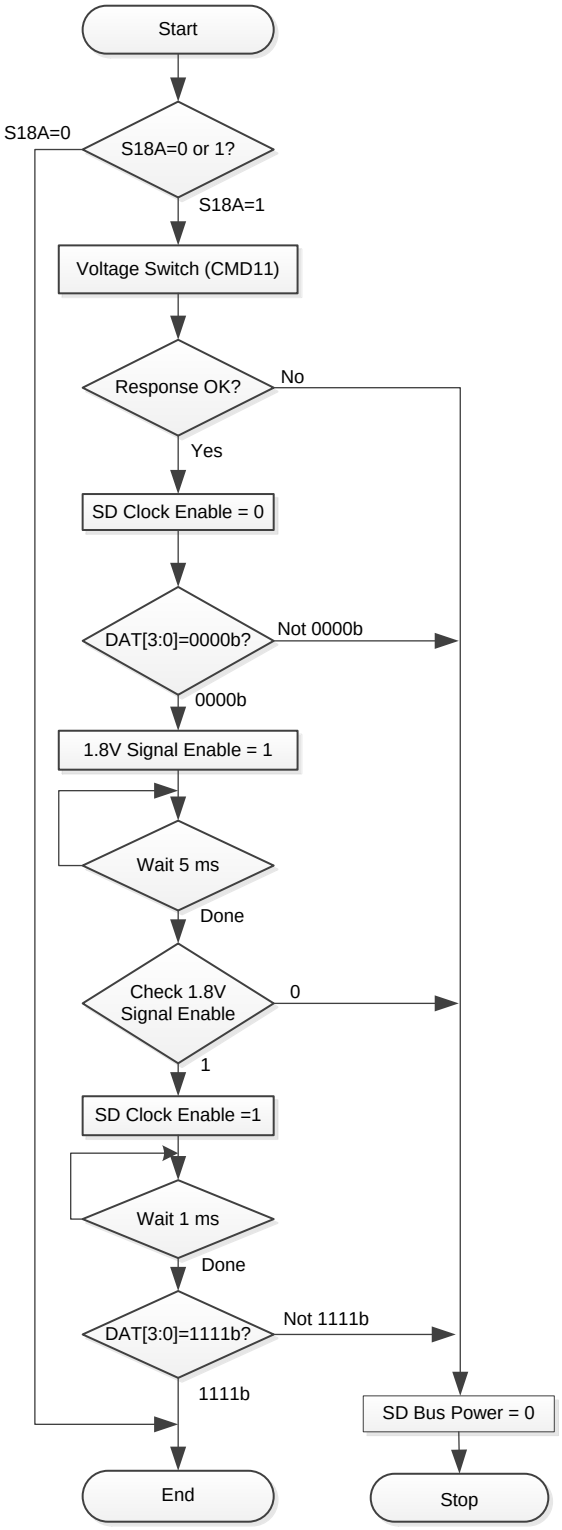
Once an SDMMC controller generates a system interrupt:

- Confirm the SDMMC System interrupt was from a voltage switch event by reading SDMMC_VENDOR_SYS_SW_CNTRL_0_VOLT_SWITCH_INT _STATUS.

33.5.7.3 Voltage Switch Procedure

The following figure shows the standard voltage switch procedure.

Figure 127: Voltage Switch Flowchart



33.5.7.1 Exit

Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_VOLT_SWITCH_INT _STATUS to clear the interrupt condition.

33.5.8 Tuning

Tuning is required because SD/MMC controllers do not automatically vary the RX clock trim each time Execute Tuning is set to 1; the SD Host driver must do this manually.

SD 3.0 cards can operate at frequencies up to 208 MHz. For the card to operate at high frequencies, the correct tap value must be programmed. The tap value in the vendor clock control register controls the track delay so that the input data is sampled correctly. The appropriate tap delay for the given platform is found using frequency tuning. The passing tap value range can be found by incrementing the tap value after each iteration and running frequency tuning.

In frequency tuning, CMD19 for SD and CMD21 for eMMC (emmc4.51) card is issued to the card and wait for the Buffer Read Ready interrupt. Once the interrupt is generated, in HOST_CONTROL_2 register (offset 03Ch), check whether EXECUTE_TUNING bit is cleared and the SAMPLING_CLK_SELECT bit is set. If both the conditions are met, treat tuning as successful and consider the tap value as working. After finding the passing tap range, the ideal tap value is calculated, which is 75% between the highest passing tap and the lowest passing tap. Set this tap value and run frequency tuning again to confirm that the tap value works. If the passing tap range falls in between 0 and 10, discount it and try to find a higher passing tap range.

During testing, the tap range between 0 and 10 was found to be insufficient for operating frequencies up to 208MHz.

33.5.8.1 Initialization

Before the standard SD drive begins running:

1. Route the SD/MMC controller's system interrupt to the appropriate driver.
2. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_TUNING_SYS_INT_STATUS to clear any latent interrupts.
3. Set SDMMC_VENDOR_SYS_SW_CNTRL_0_TUNING_SYS_INT_ENABLE to enable the system interrupt on an SD3.0 UHS-I tuning start event.

33.5.8.2 Entry

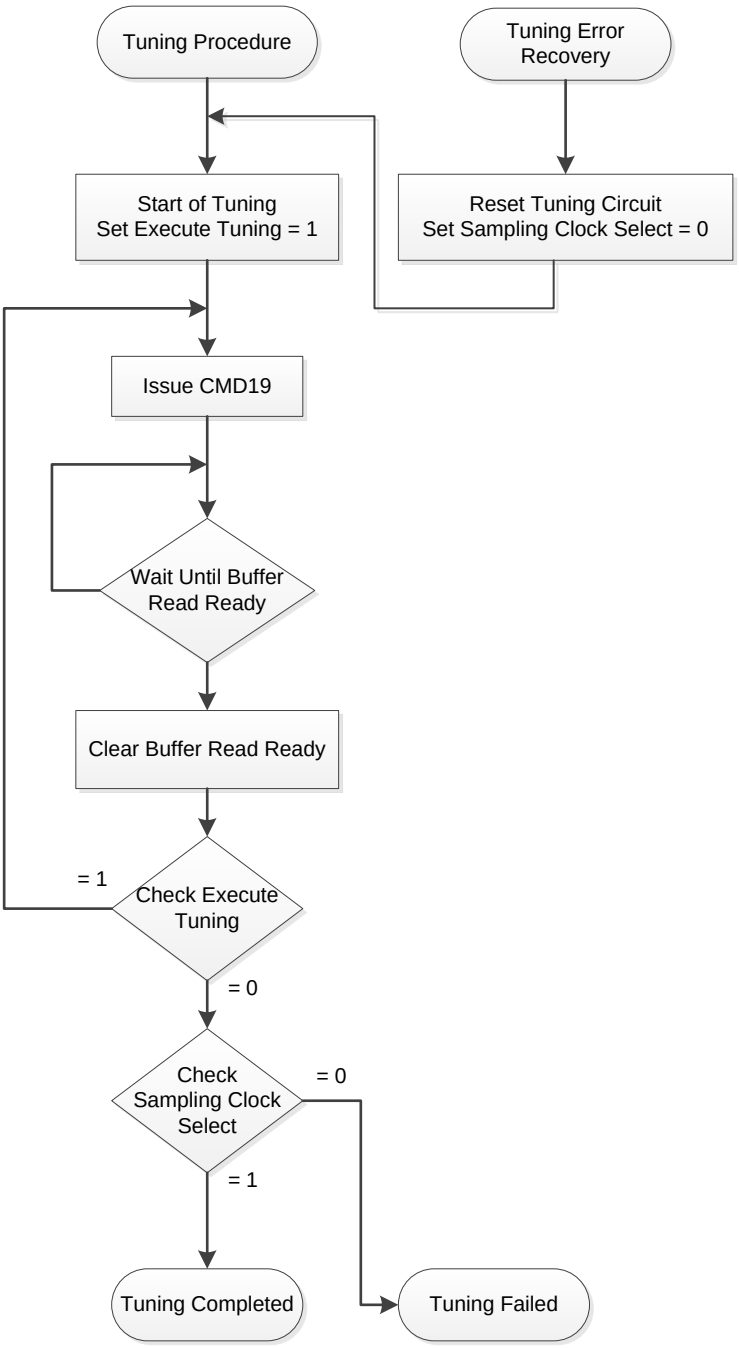
Once an SD/MMC system interrupt is received:

1. Confirm the SDMMC System interrupt was from a tuning event by reading SDMMC_VENDOR_SYS_SW__CNTRL_0_TUNING_SYS_INT_STATUS.
2. Disable the future tuning interrupts (to avoid interrupting yourself) by clearing SDMMC_VENDOR_SYS_SW_CNTRL_0_TUNING_SYS_INT_ENABLE.
3. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_TUNING_SYS_INT_STATUS to clear the interrupt condition.
4. Disable the Buffer Read Ready interrupt to keep the standard SD Host Driver inactive until the tuning sequence is complete: Clear SDMMC_INTERRUPT_STATUS_ENABLE_0_ BUFFER_READ_READY.

33.5.8.3 Tuning Procedure

The figure below shows the standard tuning procedure.

Figure 128: Tuning Procedure



33.5.8.4 Exit

1. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_TUNING_SYS_INT_STATUS to clear any pending tuning interrupts.
2. Set SDMMC_VENDOR_SYS_SW_CNTRL_0_TUNING_SYS_INT_ENABLE to enable the system interrupt on any future SD3.0 UHS-I tuning start events.
3. Signal to the SD Host driver that tuning has completed by asserting SDMMC_VENDOR_SYS_SW_CNTRL_0_ASSERT_BUFF_RD_RDY_INT.

33.5.9 SD Bus Power

When the standard SD driver changes the Power Control Register's SD Bus Power field, the hardware needs assistance from platform software to implement the change via the PMIC.

33.5.9.1 Initialization

Before the standard SD drive begins running:

1. Route the SD/MMC controller's system interrupt to the appropriate driver.
2. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_SD_BUS_POWER_ON_OFF_INT_STATUS to clear any latent interrupts.
3. Set SDMMC_VENDOR_SYS_SW_CNTRL_0_SD_BUS_POWER_ON_OFF_INT_ENABLE to enable the system interrupt on an SD3.0 UHS-I Bus Power change event.

33.5.9.2 Entry

Once an SD/MMC system interrupt is received:

1. Confirm the SDMMC System interrupt was from a Bus Power change event by reading SDMMC_VENDOR_SYS_SW_CNTRL_0_SD_BUS_POWER_ON_OFF_INT_STATUS.
2. Write 1 to SDMMC_VENDOR_SYS_SW_CNTRL_0_SD_BUS_POWER_ON_OFF_INT_STATUS to clear the interrupt condition.
3. Read SDMMC_POWER_CONTROL_HOST_0_SD_BUS_POWER to determine the power state change request.

   If Bus Power is cleared, power off the SD power supply via the PMIC. Otherwise, if Bus Power is set, power on the SD power supply via the PMIC.

33.5.9.3 Exit

No actions required.

33.6 SD/MMC Registers

Refer to "Reading Register Tables" in the Introduction section for the register table protocol as well as recommendations for accessing registers.

33.6.1 Standard Registers

Note: For standard registers' usage, refer to the SD Host Controller Specification version 3.00.

There are a few vendor-specific register fields within the standard register offset range (0x00 through 0xff). (Future products will move these vendor-specific register fields out of the standard-defined register offset range.) These fields are described below:

33.6.1.1 Present State Register

Offset: 0x24 | Read/Write: R | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
28:25	X	DAT_7_4_LINE_LEVEL: This status reflects the DAT[7:4] lines' state for debug.

33.6.1.2 Interrupt Status Register

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:30	0	VEND_SPEC_ERR Bit 1: BOOT_ACK_ERR: Occurs When Boot Ack Status is not equal to '010' Bit 0: BOOT_ACK_TIMEOUT_ERR: Occurs When Boot Ack is not received within the programmed number of cycles.

33.6.1.3 Interrupt Status Enable Register

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:30	0	VENDOR_SPECIFIC_ERR 0 = Enable neither BOOT_ACK_ERR nor BOOT_ACK_TIMEOUT_ERR statuses. 1 = Enable only BOOT_ACK_TIMEOUT_ERR status. 2 = Enable only BOOT_ACK_ERR status. 3 = Enable both BOOT_ACK_ERR and BOOT_ACK_TIMEOUT_ERR statuses.

33.6.1.4 Interrupt Signal Enable Register

Offset: 038h | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:30	0	VENDOR_SPECIFIC_ERR 0 = Enable neither BOOT_ACK_ERR nor BOOT_ACK_TIMEOUT_ERR interrupts to CPU. 1 = Enable only BOOT_ACK_TIMEOUT_ERR interrupt to CPU. 2 = Enable only BOOT_ACK_ERR interrupt to CPU. 3 = Enable both BOOT_ACK_ERR and BOOT_ACK_TIMEOUT_ERR interrupts to CPU.

33.6.1.5 Force Event Register

Offset: 050h | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:30	0	VENDOR_SPECIFIC_ERR_STATUS 0 = no effect 1 = Force a BOOT_ACK_TIMEOUT_ERR event 2 = Force a BOOT_ACK_ERR event. 3 = Force both BOOT_ACK_ERR and BOOT_ACK_TIMEOUT_ERR events.

33.6.2 Vendor Registers

33.6.2.1 SDMMC_VENDOR_CLOCK_CNTRL_0

The following registers are Vendor Specific Registers and are mapped to Vendor Specific Address Space (0x100 - 0x1FF)

Vendor Clock Control Register

TAP_VAL - Tap Value for the input data path trimmer

This determines the tap value needed to sample the input data correctly. The delay per each tap can range from 70ps to 500ps. The following values are recommended on different SD/MMC interfaces for all modes up to SDR25:

- SDMMC1 - 2
- SDMMC2 - 2
- SDMMC3 - 3
- SDMMC4 - 5

For SDR50 and above modes, the tap value is determined by the tuning procedure.

BASE_CLK_FREQ: Software should program the actual clock frequency programmed in CAR registers CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC*_0 for each SD/MMC controller. This should be done after every time SDMMC is reset and after every soft reset. This is important because all SD/MMC controllers follow the same register map, but could be programmed with different frequencies depending on the use case.

SPI_MODE_CLKEN_OVERRIDE: Software should always program this to 0 (NORMAL).

PADPIPE_CLKEN_OVERRIDE: Software should always program this to 0 (NORMAL).

Offset: 0x100 │ Read/Write: R/W │ Reset: 0x0005d00d (0b00000000000101110100000xx0x1101)

Bit	Reset	Description
28:24	0x0	TRIM_VAL: Reserved. Software should not change these bits.
23:16	0x5	TAP_VAL: Tap value for input data path trimmer.
15:8	0xd0	BASE_CLK_FREQ: System software programs the base SD/MMC clock frequency into this register before loading the SD/MMC driver. This value is readable in the standard, but not writeable, SDMMC_CAPABILITIES_0_BASE_CLOCK_FREQUENCY register field.
5	0x0	SDR50_TUNING_OVERRIDE: override the SDR50_TUNING capabilities bit. Software should only set this bit if it is required to use Tuning for SDR50. (only supported for SDMMC1 and SDMMC3) 0 = NORMAL : 0 -> No tuning support advertised for SDR50 mode. 1 = OVERRIDE: 1 -> Tuning support is enabled for SDR50 mode.
3	0x1	PADPIPE_CLKEN_OVERRIDE: Override for pad macro and pipe macro clken. 0 -> CLKEN is to pad macro and pipe macro can be deasserted along with internal CLKEN. 0 = NORMAL: 0 -> CLKEN is to pad macro and pipe macro can be deasserted along with internal CLKEN. 1 = OVERRIDE: 1 -> CLKEN is kept asserted to pad macro and pipe macro when internal CLKEN is deasserted.
2	0x1	SPI_MODE_CLKEN_OVERRIDE: Override for CLKEN during SPI_MODE during sw_reset. 0 = NORMAL: 0 -> CLKEN is deasserted while doing sw_reset. 1 = OVERRIDE: 1 -> CLKEN is kept asserted while doing sw_reset.
1	0x0	INPUT_IO_CLK: Feedback clock is selected by default. Software should not change this. Disabling the Feedback clock will select the Internal Clock that requires different tap value programming. 0 = FEEDBACK 1 = INTERNAL
0	0x1	SDMMC_CLK: This is set when sdmmc_clk is supplied by the CAR module. Prior to sdmmc_clk switch OFF. This bit should be written '0'. By writing zero, the asynchronous card interrupt is routed to the Interrupt controller.

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE

33.6.2.2 SDMMC_VENDOR_SYS_SW_CNTRL_0

Vendor System SW Control Register

Offset: 0x104 │ Read/Write: R/W │ Reset: 0xX0018000 (0bxxxxxxxxxxxxxxxx11000xxxxx0000x00)

Bit	R/W	Reset	Description
31:28	RO	X	DAT_7_4_LINE_LEVEL: Line Signal Level. This status is used to check the DAT line level to recover from errors, and for debugging.
16	RW	0x1	SD_FENCE_RD_EN_AT_BLKGAP: set this to generate fence read at the blkgap ('stop at blkgap' should be enabled before setting this) 0 = DISABLE 1 = ENABLE
15	RW	0x1	SD_FENCE_RD_ENABLE: Fence Read request enable - need to set this to avoid data coherency issues when writing to system memory 0 = DISABLE 1 = ENABLE
14	RW	0x0	SD_BUS_POWER_ON_OFF_INT_STATUS: SD_BUS_POWER was changed. System software can use this interrupt to implement power switch. 0 = NO_INT 1 = GEN_INT
13	RW	0x0	VOLT_SWITCH_INT_STATUS: VOLT_18_EN was changed. System software can use this interrupt to implement a UHS-I voltage switch procedure for a standard SD Host driver 0 = NO_INT 1 = GEN_INT
12	RW	0x0	TUNING_SYS_INT_STATUS: CMD19 was issued while EXECUTE_TUNING was set. System software can use this interrupt to implement a UHS-I tuning procedure for a standard SD Host driver. 0 = NO_INT 1 = GEN_INT
6	RW	0x0	SD_BUS_POWER_ON_OFF_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when SD_BUS_POWER is changed. 0 = DISABLE 1 = ENABLE
5	RW	0x0	VOLT_SWITCH_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when VOLT_18_EN is changed. 0 = DISABLE 1 = ENABLE
4	RW	0x0	TUNING_SYS_INT_ENABLE: Enables a separate system interrupt (i.e., not the standard SDHCI interrupt) when CMD19 is issued while EXECUTE_TUNING is set. 0 = DISABLE 1 = ENABLE
3	RW	0x0	ASSERT_BUFF_RD_RDY_INT: Write a 1 to this field to assert sdmmc_interrupt_status_0_buffer_read_ready. Used by the system software that implements the tuning procedure to signal to the standard SD driver that the tuning process has completed 0 = DISABLE 1 = ENABLE
1	RW	0x0	INT_MASK_WHILE_TUNING: to avoid breaking software compatibility, interrupt generation behavior is changed when a this bit is set 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
0	RW	0x0	SPI_MODE: This is a mirror bit. The SPI mode is set if this bit is set or CMD_XFER_MODE[7] is set. Writing 1 will drive the CS Low and writing zero will deassert the CS signal 0 = DISABLE : 0 -> SPI mode is disabled. 1 = ENABLE: 1 -> SPI mode is enabled.

33.6.2.3 SDMMC_VENDOR_CAP_OVERRIDES_0

Capabilities Override Bits

Offset: 0x10c │ Read/Write: R/W │ Reset: 0x00000007 (0b111)

Bit	Reset	Description
2	0x1	VOLTAGE_3_3_V_SUPPORT_OVERRIDE: Voltage support 3_3_V override
1	0x1	VOLTAGE_3_0_V_SUPPORT_OVERRIDE: Voltage support 3_0_V override
0	0x1	VOLTAGE_1_8_V_SUPPORT_OVERRIDE: Voltage support 1_8_V override

33.6.2.4 SDMMC_VENDOR_BOOT_CNTRL_0

Vendor Boot Control Register

This register is used to configure Boot Mode to support MMC v4.3 cards.

Offset: 0x110 │ Read/Write: R/W │ Reset: 0x00000000 (0b00)

Bit	Reset	Description
1	0x0	BOOT_ACK: This bit is used to support Boot Option in MMC 4.3 version cards. If set Boot acknowledgment is given by card else not given by card 0 = DISABLE 1 = ENABLE
0	0x0	BOOT: This bit enables/disable BootOption1. If set BootOption1 is enabled, hardware auto clears it when boot data is done. Writing 0 terminates the BootOption1 0 = DISABLE 1 = ENABLE

33.6.2.5 SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0

Vendor Boot Acknowledgment Timeout Register

Offset: 0x114 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000)

Bit	Reset	Description
19:0	0x0	VALUE: If Boot Acknowledgment is not received within the programmed number of cycles, a Boot Acknowledgement Timeout error occurs (VENDOR_SPECIFIC_ERR[0])

33.6.2.6 SDMMC_VENDOR_BOOT_DAT_TIMEOUT_0

Boot Data Timeout Register

Offset: 0x118 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000)

Bit	Reset	Description
24:0	0x0	VALUE: If Boot Data is not received within the programmed number of cycles, a Data Timeout error occurs.

33.6.2.7 SDMMC_VENDOR_DEBOUNCE_COUNT_0

Debounce Counter Value Register

The Debounce Counter runs on a 32 KHz clock. Keeping the default value to 100ms = (100 * 32cycles/1ms) = 3200 cycles for 100ms = 0xC80

Offset: 0x11c │ Read/Write: R/W │ Reset: 0x00000c80 (0b00000000000000110010000000)

Bit	Reset	Description
23:0	0xc80	VALUE: The number of 32KHz clock cycles is programmed to meet the debounce period of the card slot.

33.6.2.8 SDMMC_VENDOR_MISC_CNTRL_0

Miscellaneous Vendor Control Register

- SDMMC_SPARE0: Spare register bits with reset value of 0
  - SDMMC_SPARE0[0]: SW_RESET_CLKEN_OVERRIDE, override the ‘sdmmc_clken’ when doing SW_RESET if set to 1.
  - SDMMC_SPARE0[1]: When set, allows SD clock to be stopped in the middle of a read data block while in SDR104 mode. Unsafe for at least some SD cards, but may improve SDR104 DMA read performance in some cases.
  - SDMMC_SPARE0[2]: When set, SDR104 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_SDR104
  - SDMMC_SPARE0[3]: When set, SDR50 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_SDR50
  - SDMMC_SPARE0[5]: When 0, masks the pad macro's "high speed" enable to 0, causing the pad macro to always launch data on the falling edge of the clock. This prevents the SD Host driver's setting of SDMMC_POWER_CONTROL_HOST_x_HIGH_SPEED_EN from undesirably affecting the output timing.
  - SDMMC_SPARE0 [7:6]: Number of pipe stages.
  - SDMMC_SPARE0[8]: When set, DDR50 support is advertised in SDMMC_CAPABILITIES_HIGHER_0_DDR50
  - SDMMC_SPARE1: Spare register bits with reset value of 1
  - SDMMC_SPARE1[0]: CMD_CONFLICT_DISABLE, disable command line conflict if set to 1.

Offset: 0x120 │ Read/Write: R/W │ Reset: 0xffff0098 (0b11111111111111110000000010011000)

Bit	Reset	Description
31:16	0xffff	SDMMC_SPARE1: Spare register bits with reset value of 1
15:1	0x4c	SDMMC_SPARE0: Spare register bits with reset value of 0x40
0	0x0	ERASE_TIMEOUT_LIMIT: Erase timeout value. 0 = FINITE: Finite, It is limited to the programmed value in the DATA_TIMEOUT_VALUE 1 = INFINITE: Infinite, Controller would be monitoring until the card is busy.

33.6.2.9 SDMMC_MAX_CURRENT_OVERRIDE_0

Offset: 0x124 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000)

Bit	Reset	Description
23:16	0x0	OVERRIDE_FOR_1_8V: Maximum override for 1.8V
15:8	0x0	OVERRIDE_FOR_3_0V: Maximum override for 3.0V
7:0	0x0	OVERRIDE_FOR_3_3V: Maximum override for 3.3V

33.6.2.10 SDMMC_SDMEMCOMPPADCTRL_0

SDMEMCOMP Pad control register

If AUTO_CAL_ENABLE is disabled (0), the values in this register are used to drive the DRVUP/DRVDN controls to the SDMMC comp pads.

Offset: 0x1e0 │ Read/Write: R/W │ Reset: 0x29318007 (0b01010010011x0011000000000000111)

Bit	Reset	Description
30:29	0x1	PU_PAD_DRV_TYPE
28:27	0x1	PD_PAD_DRV_TYPE
26:20	0x13	CFG_SDMEMCOMP_DRVUP: used if AUTOCAL is disabled
18:12	0x18	CFG_SDMEMCOMP_DRVDN: used if AUTOCAL is disabled
11	0x0	PU_PAD_E_TEST_OUT
10	0x0	PD_PAD_E_TEST_OUT
9:7	0x0	PU_PAD_TEST_SEL
6:4	0x0	PD_PAD_TEST_SEL
3:0	0x7	SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL

33.6.2.11 SDMMC_AUTO_CAL_CONFIG_0

SDMEMCOMP pad auto-calibration settings - Valid for SDMMC1/SDMMC3 Instances

AUTO_CAL_SLW_OVERRIDE

- 0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output DRDVDN/UP_SLWR/F[1:0] = AUTO_CAL_PULLDOWN/UP[4:3]
- 1 (override) use CFG2TMC_SDIO[1|3]*_DRVDN/UP_SLWR/F pins to control pad slew inputs

AUTO_CAL_OVERRIDE

- 0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting
- 1 (override) : use AUTO_CAL_PU/PD_OFFSET register values directly

Offset: 0x1e4 │ Read/Write: R/W │ Reset: 0x00010000 (0b0000xxxxxxxxx001x0000000x0000000)

Bit	Reset	Description
31	0x0 –	AUTO_CAL_START: Writing a one to this bit starts the calibration state machine. This bit must be set even if the override is set in order to latch in the override value.
30	0x0	AUTO_CAL_OVERRIDE: AUTOCAL override. 0 = NORMAL : 0 (normal operation): use AUTO_CAL_PU/PD_OFFSET as an offset to the calibration state machine setting 1 = OVERRIDE : 1 (override) : use AUTO_CAL_PU/PD_OFFSET register values directly
29	DISABLED	AUTO_CAL_ENABLE: AUTOCAL enable. 0 = DISABLED : 0 (disabled): use sdmmc2tmc_cfg* register settings for pull-up/dn 1 = ENABLED : 1 (normal operation): use SDMMC generated pull-up/dn (override or AUTOCAL)
28	0x0	AUTO_CAL_SLW_OVERRIDE: AUTOCAL slew rate override. 0 = NORMAL : 0 (Normal operation) pad DRVDN/UP_SLWR/F tied to AUTO_CAL output DRDVDN/UP_SLWR/F[1:0] = AUTO_CAL_PULLDOWN/UP[4:3] 1 = OVERRIDE : 1 (override) use CFG2TMC_SDIO[1|3]*_DRVDN/UP_SLWR/F pins to control pad slew inputs
18:16	0x1	AUTO_CAL_STEP: calibration step interval (in microseconds)

Bit	Reset	Description
14:8	0x0	AUTO_CAL_PD_OFFSET: 2's complement offset for pull-down value
6:0	0x0	AUTO_CAL_PU_OFFSET: 2's complement offset for pull-up value

33.6.2.12 SDMMC_AUTO_CAL_INTERVAL_0

SDMEMCOMP Pad Calibration Interval

Offset: 0x1e8 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	AUTO_CAL_INTERVAL: 0: Do calibration once. Otherwise, auto-calibration occurs at intervals equivalent to the programmed number of microseconds.

33.6.2.13 SDMMC_AUTO_CAL_STATUS_0

SDMEMCOMP Pad Calibration Status

Offset: 0x1ec │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	AUTO_CAL_ACTIVE: One when auto calibrate is active - valid only after auto calibrate sequence has completed (AUTO_CAL_ACTIVE == 0)
30:24	X	AUTO_CAL_PULLDOWN_ADJ: Pull-down code sent to pads
22:16	X	AUTO_CAL_PULLUP_ADJ: Pull-up code sent to pads
14:8	X	AUTO_CAL_PULLDOWN: Pull-down code generated by auto-calibration
6:0	X	AUTO_CAL_PULLUP: Pull-up code generated by auto-calibration

[THIS PAGE INTENTIONALLY LEFT BLANK]

34.0 MIPI-HSI (HIGH SPEED SYNCHRONOUS SERIAL INTERFACE)

The MIPI High Speed Synchronous Serial Interface (“HSI”) connects a mobile application processor with the cellular baseband modem. It is a full duplex, point-to-point interface providing peer-to-peer communication between the two devices. The HSI provides for up to 16 channels of data transfer. The 16 channels share the same single flow control, so they cannot pass each other. This means different qualities of service (e.g., isochronous vs. low latency vs. bandwidth soak) are managed at a higher level.

In each direction, the HSI uses four wires to communicate a single bit of data:

- FLAG
- DATA
- READY
- WAKE

The data bit and clock are encoded into two of the wires, FLAG and DATA. These signals do not toggle when no data is being transferred. Although this complicates the receiver, it significantly reduces power compared to an interface that requires a continuous clock. READY and WAKE are used for flow control. WAKE signals the RX side that data transfer may be pending. READY signals the TX side when it may send data.

34.1 MIPI Standards Support

The MIPI HSI Controller implements the following MIPI standards:

- *High Speed Synchronous Serial Interface Specification, Version 1.0*, approved 23 March 2004
- *High-speed Synchronous Serial Interface (HSI) Physical Layer, Version 1.01.00*, approved 25 January 2009

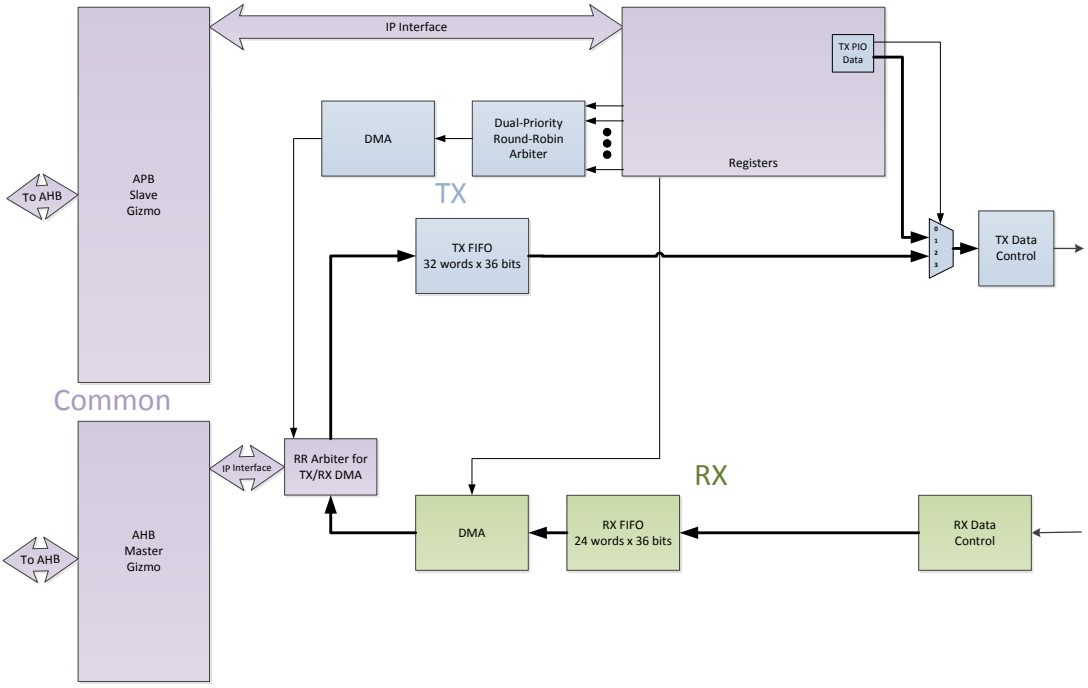
Refer to the MIPI website for further details of these specifications.

34.2 Overview and Architecture

The HSI Controller is a master on the AHB bus, using the standard AHB interface (Gizmo) for DMA to and from main memory. The register programming and PIO interface is a slave path on APB. The HSI controller implements TX and RX functions completely independently, so that both the TX and the RX paths can be active at the same time. The controller supports 16 TX and 16 RX channels. The driver software can setup additional DMA channels while one or more channels are already active. As both TX and RX are running simultaneously, DMA can have requests from both. A round robin arbiter is used to arbitrate the requests between TX and RX. Software has flexibility to provide the priority of channels. TX implements the PIO path as well which carries higher priority than any other DMA channel.

The transmitter logic runs at the TX clock provided from the system clock controller. The receiver has to retrieve its clock from the DATA and FLAG signals coming in from an external transmitter. The method to retrieve the receiver clock is explained in the Receiver section. The controller implements separate FIFOs for TX and RX.

Figure 129: HSI Controller Top-Level Block Diagram



34.2.1 DMA Functionality

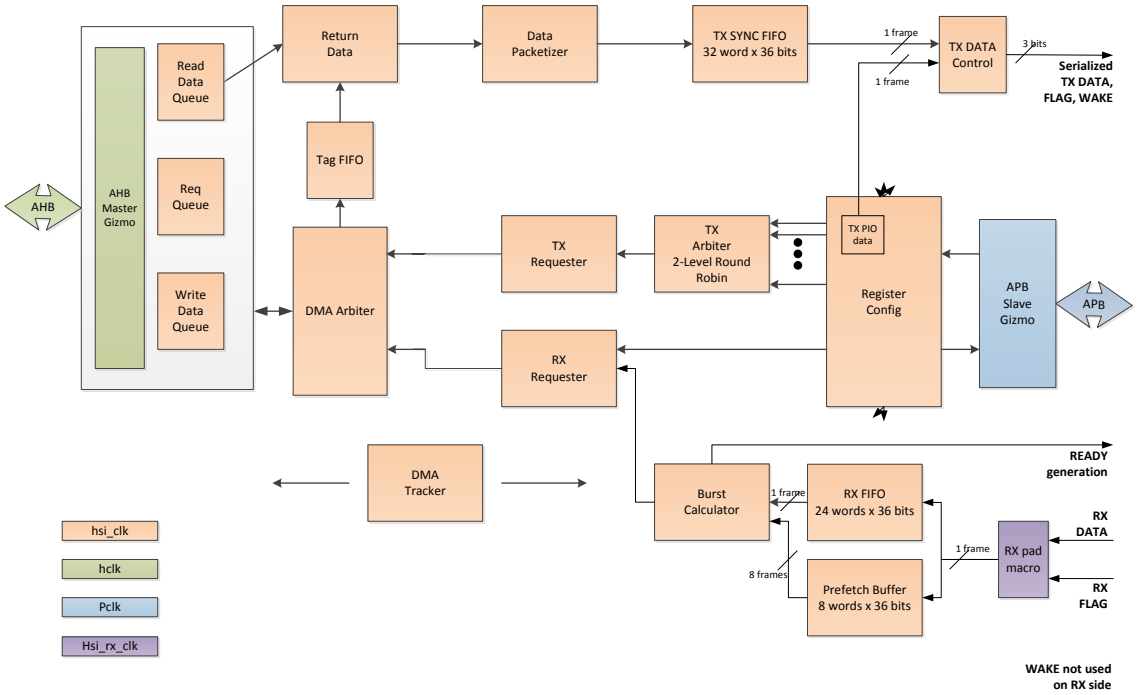
The HSI Controller is a master on AHB bus, using DMA to read HSI TX data and write HSI RX data. When both TX and RX DMA requests are pending internally, they are arbitrated using round-robin arbitration.

The controller implements a two-level (high and low priority) round robin arbitration to choose one channel out of 16 TX channels. The tag FIFO keeps track of AHB return data's HSI TX channel to push the channel information along with the data into the TX FIFO.

On the RX side, the RX Requester will coalesce successive frames into a single burst, up to the maximum allowed burst size. So, to maximize AHB write-to-system memory efficiency, the baseband modem should switch channels as little as possible.

DMA Tracker keeps track of all DMA TX and RX transactions.

Figure 130: HSI Controller Internal Micro Architecture



34.2.2 Transmitter Data Flow

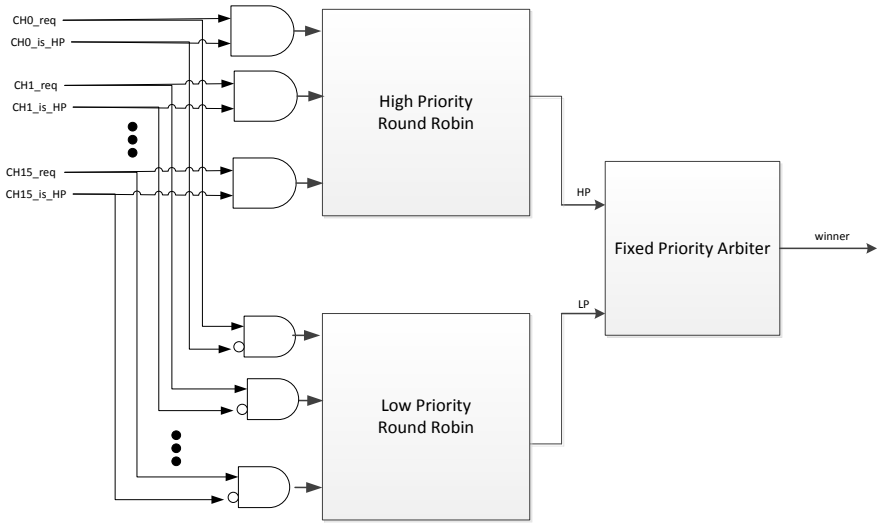
Software can configure up to 16 DMA channels to transmit data. A DMA channel can be set up, started, and stopped, even while other channels are actively DMA'ing. Setting up a DMA channel requires system memory address, DMA length, maximum burst size, and priority of that channel. To avoid bubbles between completing one DMA buffer and starting a new one, the HSI controller provides 2 DMA buffers per channel. That way, the HSI driver can set up the next DMA buffer for a channel while the other buffer can still service HSI traffic in that channel.

The HSI controller implements a two-level round robin arbiter to arbitrate between 16 TX channels. Re-arbitration occurs when the previous burst completes.

34.2.2.1 TX Two-Level Arbiter

The two level arbiter is implemented in such a way that High Priority channels always get through first and potentially starve low priority channels. The HSI host driver has the responsibility to avoid starving low-priority channels. Software can choose either priority for each channel independently. High priority channels are passed through a simple round-robin arbiter and similarly low-priority channels are also passed through a round robin and then the output of these two arbiters are given to a fixed priority arbiter which always make sure that HP goes out first.

Figure 131: Two-Level Arbiter used for TX



34.2.3 Receiver Data Flow

Software should set up a DMA buffer for each channel that may have RX data. For any data received in a channel without a DMA buffer allocated, the HSI controller:

- Interrupts software that an unexpected channel was received, and
- Halts further data (by deasserting READY) until software has drained the unexpected data

When allocating RX data buffers, software may not know the length of data expected ahead of time.

In case of unknown DMA length, controller generates the “new data write completion” interrupt on every word/burst completion (when successfully written to system memory). That way software knows when new data (of previously unknown length) has arrived.

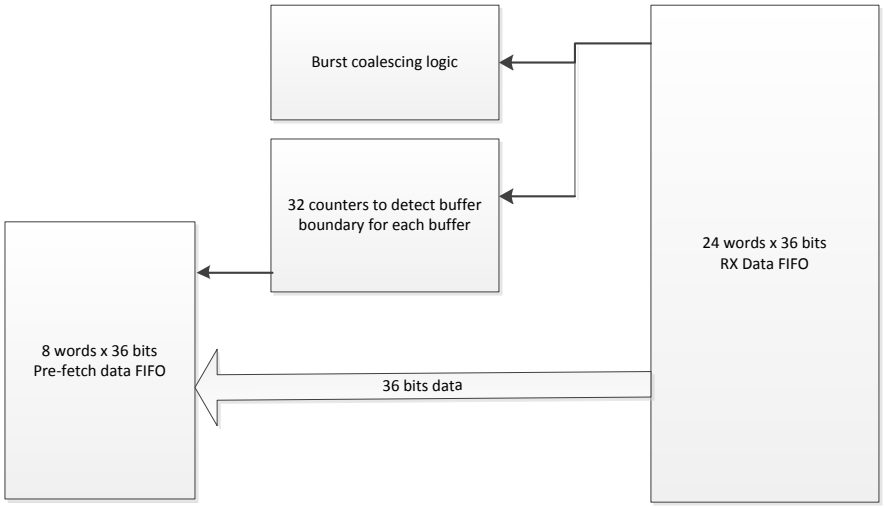
34.2.3.1 Receiver Burst Coalescing

Burst coalescing needs to be implemented to effectively use the system bus. Data coming in from external RX can be in any order in terms of channel number. The controller needs to make the maximum possible burst of incoming data before putting a request onto system bus.

As data is popped out of the RX data FIFO, its channel is compared against the next FIFO entries’ channels. If successive FIFO entries (HSI frames’ data) share the same channel, they will be coalesced into a single burst, up to the maximum burst size. (For AHB, the maximum burst size is eight 32-bit words.) Thus, AHB and system memory efficiency are strongly affected by RX data channel ordering.

The controller provides flexibility to software to switch between two system buffers in the runtime. Burst coalescing is invalid when software wants to switch the write location in the system memory. In that case controller needs to keep track of data length received associated for a particular buffer. The controller maintains counters for each buffer to detect the buffer boundary. The figure below shows the top level view of the burst coalescing scheme.

Figure 132: Burst Coalescing Scheme



34.3 Functional Description

34.3.1 Receiver Data Flow Modes

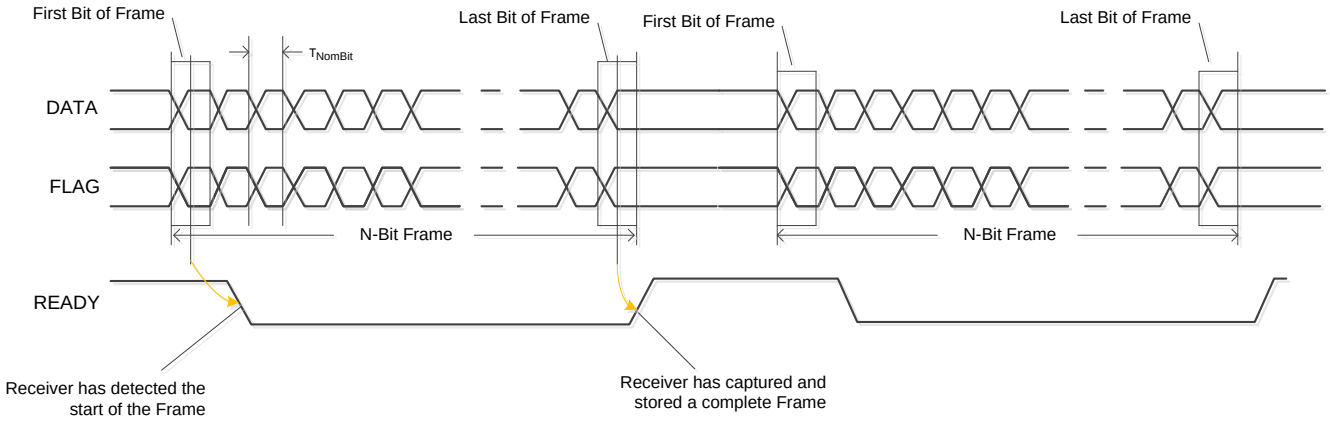
Data flow is a receiver feature. There are three different data flow types depending on the READY behavior -

34.3.1.1 Synchronized Data Flow

In the synchronized data flow, the receiver shall set READY to logic zero upon the reception of the first bit of frame and shall not set READY to logic one until enough bits have been received for the frame, and the frame has been stored in the buffer. The figure below shows the synchronized data flow READY signal timing.

Because of the idle time introduced by this handshake, synchronized data flow has lower interface efficiency, and should be avoided unless necessary.

Figure 133: Synchronized Data Flow Timing



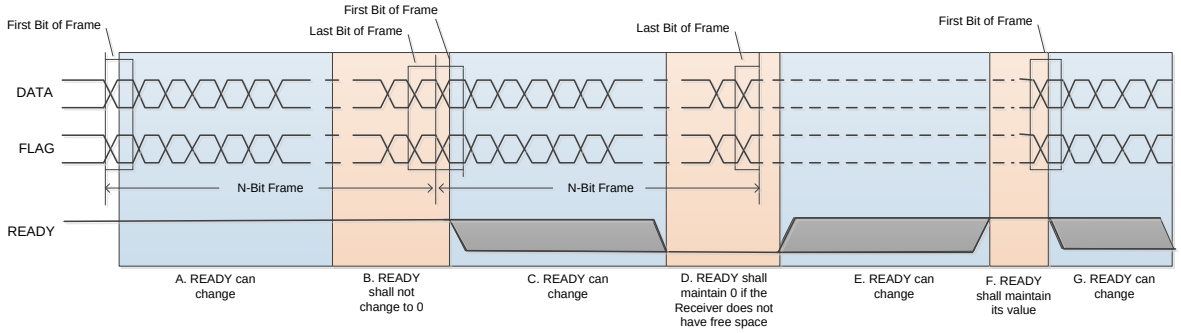
34.3.1.2 Pipelined Data Flow

In the pipelined data flow received shall only set the READY signal low when it has no storage available for a new frame. As long as the receiver can guarantee the successful reception of at least one more frame following the one being received, it

shall set READY to logic 1, and transmission can proceed in a pipelined manner. Pipelined data flow enables back-to-back frame transmission with no idle period on the line.

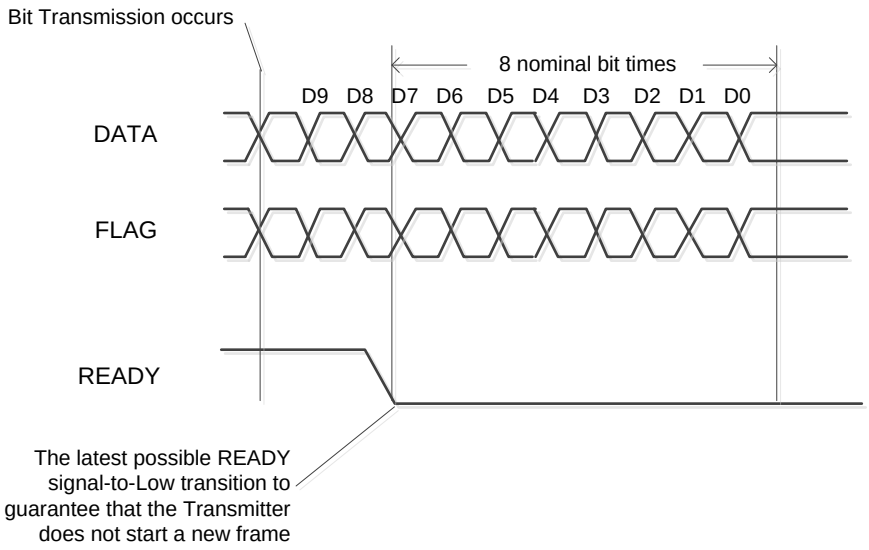
Because pipelined data flow allows for the highest performance, it should be used whenever possible.

Figure 134: Pipelined Data Flow Timing



The timing of the READY signal is asynchronous with respect to DATA and FLAG. However, the receiver shall drive READY to logic zero at least eight nominal bit times before the end of current frame in order to delay transmission of the next frame.

Figure 135: Pipelined Data Flow READY Signal-to-Low Detail

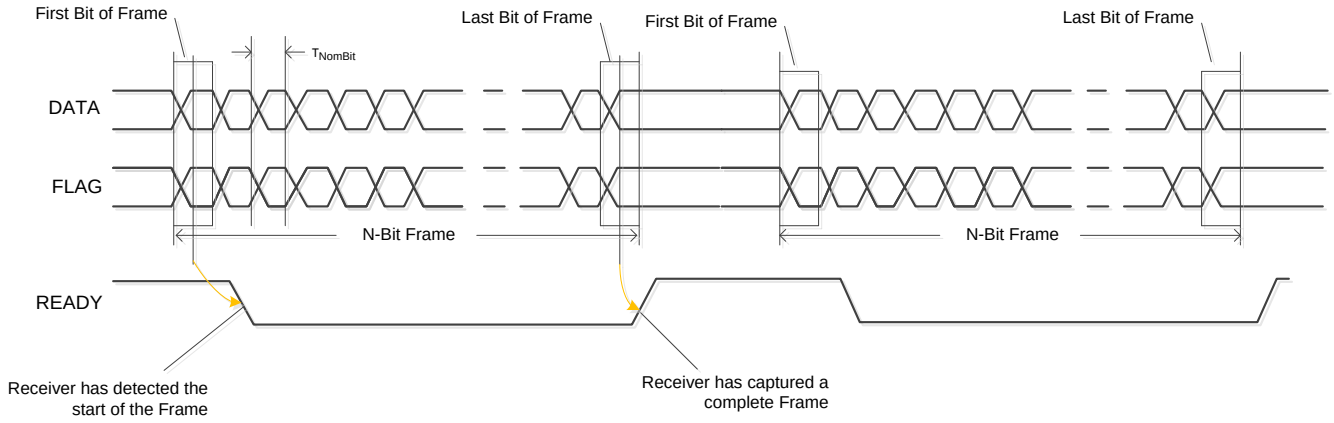


34.3.1.3 Receiver Real-time Data Flow

In receiver real-time data flow, a receiver shall drive READY to logic zero after the first bit of frame is found and high when the last bit of the frame is found. This READY signal behavior enables faster data transmission than in the synchronized data flow since the idle period between the frames are shorter.

Real-time data flow is used only for error recovery. (It provides a way for the TX side to see where in the frame the RX side is.) It should not be used for normal data transmission.

Figure 136: Receiver Real-Time Data Flow Timing



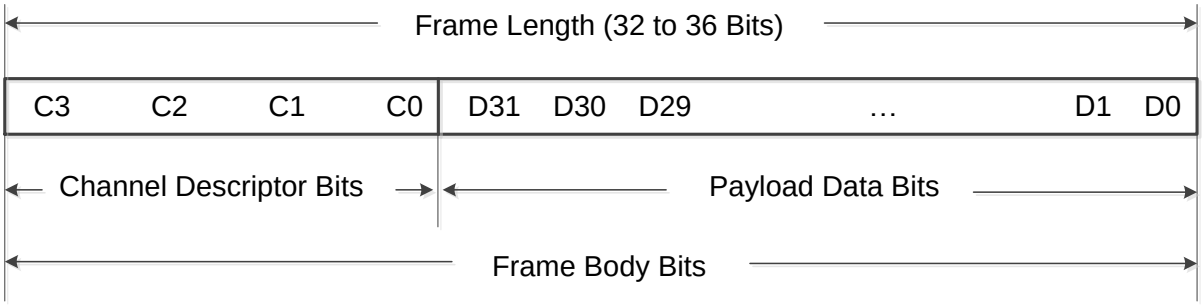
34.3.2 Bit Transmission Mode

HSI supports two transmission modes: Stream transmission mode and Frame transmission mode. The HSI receiver and transmitter implement Stream transmission mode with synchronized data flow, and Frame transmission mode with Synchronized Data Flow and Pipelined Data Flow.

34.3.2.1 Stream Transmission Mode

In this mode, the Physical layer data frame structure is made of 32 payload data bits, and 0 to 4 channel descriptor bits as shown below.

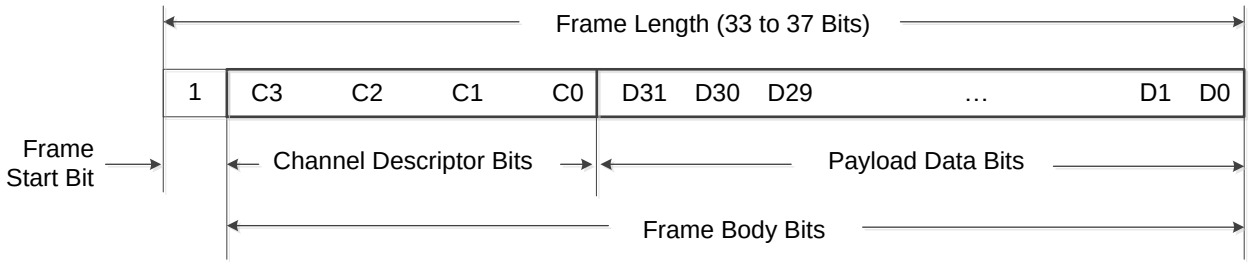
Figure 137: Frame Structure in Stream Mode



34.3.2.2 Frame Transmission Mode

In this mode, Frame boundaries shall be defined by a Frame start bit and a number of frame body bits.

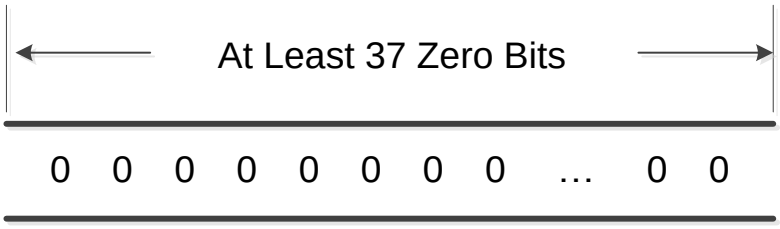
Figure 138: Frame Structure in Frame Mode



34.3.2.3 Physical Layer Break Frame

The transmitter shall be able to send a break frame regardless of the state of READY signal. Formally, the break transmission shall be defined as any sequence of logic zeros that is longer than can occur in normal transmission.

Figure 139: Minimum Break Transmission Size for Four Channel Descriptor Bits



34.4 Programming Guidelines

34.4.1 Transmitter

1. Program TX_MODE and tx channel_width in the TX_Global_Control_Register. Channel_width decides the Frame length. For example,if software wants to use up to 8 channels, channel_width must be programmed as 3. Software can use channel number 0 to 7 in this case.
2. Software can configure up to 2 DMA Buffers per channel. Provide the DMA length, Priority, and Starting address for each buffer and enable ready/in-use bit in TX_Control_Register for respective channel. Hardware will finish the running buffer first and then switch to the next buffer of same channel if that buffer is enabled.
3. Enable Buffer Complete Interrupt in TX_Buffer_Complete_Interrupt_Enable_Register. Hardware asserts the interrupt when total length for the corresponding buffer is transferred. Interrupt for the corresponding channel will be generated in TX_Buffer_Complete_Interrupt_Status_Register.
4. Software can initiate High priority PIO mode at any time. Program channel number in TX_Global_Control_Register through which PIO frame has to be transmitted, write 32 bit data in PIO_Word_Register. Writing data into this register initiates the PIO transfer. The next PIO write cannot be made until the TX PIO Complete Interrupt (“PIO_COMPLETE_INTR”) asserts.
5. Software can see the status of each buffer in TX_Buffer_Status_Register (shows how many words have been transmitted to TX) and can HALT any channel if need. **Hardware will clear the Ready/in-use bit in TX_Control_Register for that Buffer on Halt.** On Halt, DMA will stop fetching further data from that buffer.
6. Break frame feature is supported to get TX and RX in sync. Software can enable the TX to send break frame anytime (but only in frame mode). On asserting break frame, the controller will send a frame of all zeros (the frame length will depend on channel_length).

34.4.2 Receiver

1. Program the number of channels and enable the RX_ENABLE bit in RX_Global_Control_Register.
2. Software can set up up to 2 DMA Buffers per channel. Software needs to program the DMA Length (even if the total DMA length is not known ahead of time) and starting address for each buffer and enable/in-use bit in RX_control_register for respective channel.
3. Not setting up DMA will result in an Error Interrupt in RX_Error_Interrupt_Status_Register when a Frame is received by the RX for that particular channel. The received data can be read from the HSI RX Error Data register.
4. Enable “New Data Write Completion Interrupt” in RX_New_Data_Wr_Comp_Interrupt_Enable Register which will generate an interrupt every time a frame (OR maximum possible bursts of a frame) will be written to system memory. Interrupt for corresponding channel will be generated in RX_New_Data_Wr_Comp_Interrupt_Status_Register.
5. Enable Buffer Complete Interrupt in RX_Buffer_Complete_Interrupt_Enable_Register. Hardware asserts the interrupt when the total length for the corresponding buffer is transferred. Interrupt for the corresponding channel will be generated in RX_Buffer_Complete_Interrupt_Status_Register. In case of fence_enable, the controller waits until fence read data comes back before asserting the RX_Buffer_Complete_Interrupt. However when fence is disabled, as soon as data is written on AHB, controller will assert the interrupt.
6. Software can see the status of each Buffer in RX_Buffer_Status_Register (shows how many words have been successfully written to system memory) and Halt any channel if need. Hardware will clear Ready/in-use bit in RX_Control_Register for that Buffer on Halt. On Halt, the RX will not write data to system memory for that buffer and generates an Error Interrupt in RX_Error_Interupt_Register if any Frame is received for that Halted channel.

34.4.3 Halting Incomplete DMA

If desired, software could de-allocate an RX buffer that is only partially filled by:

- Asserting that buffer’s “halt” bit (need to add halt bits for both RX and TX buffers, not just TX)
- Polling on the semaphore until hardware releases it

Software can know how much data is in each buffer by:

- Looking at the per-buffer “# words written” status
- Knowing that the hardware will work on the buffer given to it first, first

This requires software to serially allocate DMA buffers to the same channel (i.e., two separate threads cannot simultaneously allocate DMA buffers to the same channel).

34.4.4 Miscellaneous Points

- The driver needs to make sure of the ratio between core clock and Rx clock for successful data reception. 2:1 is the ratio that needs to be maintained between core and Rx clock. For example, if Rx is running at 200 MHz, core must be running at minimum 100 MHz for successful reception.
- READY deassertion in synchronized Data Flow mode. READY deassertion behavior is under-constrained and the description only says “upon reception of the first bit of a physical frame”. This sequence of events cannot happen in zero time. Maximum deassertion delay is not given. No reasonable implementation could allow the delay to be longer than a frame time, and a reasonable maximum would be the 8 nominal bit times before the end of a frame used by the Pipelined Data Flow. Regardless, Synchronize Data Flow is a huge efficiency waste on the HSI interface, and any high-speed, low-power device should be using Pipelined Data Flow instead.
- The Tegra® 4 MIPI-HSI Controller does not implement following:
  - Receiver Frame Burst Counter
  - Receiver Tailing bit Counter
  - Missed clock cycle

- Additional Clock Cycle

The main reason for not implementing these error detection primitives is that the HSI environment is relatively error free and error detection is quite well and recovery time is relatively long. HSI relies on the physical interface's bit error rate to be significantly small for reliability.

- In case of an RX break frame – which is all zero frames, the controller asserts the RX Error interrupt for channel 0 if the ready bit for channel 0 is not set (which means channel 0 is not running) and also asserts RX break frame interrupt.
- The Tegra 4 HSI architecture does not support fence mechanism. Keep it disabled and apply the WAR for successful operation.
- Software should deassert the HSI_RX_SRC_RST bit of MIPIHSI_GLOBAL_RX_CONTROL_0 register before starting the transfer. This is needed to release the reset of rx_pad_macro.

34.5 MIPI-HSI Registers

Refer to "Reading Register Tables" in the Introduction section for the register table protocol as well as recommendations for accessing registers.

34.5.1 MIPIHSI_GLOBAL_TX_CONTROL_0

HSI Global TX Control Register

Offset: 0x0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx00000x0000000)

Bit	Reset	Description
12:9	0x0	PIO_CHANNEL_NUM: Baseband channel number to which PIO data has to be transmitted. As writing the PIO Data Initiates the PIO word transfer, this field must be written before PIO data. TX supports data transfer through PIO as well. The controller will insert the PIO frame after the current frame on HSI interface is done and generate the PIO done interrupt. 0 = CHANNEL_0 1 = CHANNEL_1 2 = CHANNEL_2 3 = CHANNEL_3 4 = CHANNEL_4 5 = CHANNEL_5 6 = CHANNEL_6 7 = CHANNEL_7 8 = CHANNEL_8 9 = CHANNEL_9 10 = CHANNEL_10 11 = CHANNEL_11 12 = CHANNEL_12 13 = CHANNEL_13 14 = CHANNEL_14 15 = CHANNEL_15
8	0x0	SEND_BREAK_FRAME: Injects Break Frame. Possible only in Frame Transmission Mode. Hardware clears it when a Break Frame is sent out. 0 = DISABLE 1 = ENABLE
6:3	0x0	CHANNEL_DESC_WIDTH: This defines the channel_width in the frame_length. The Channel descriptor field in the Frame will change accordingly. This field must be programmed correctly before any transfer starts. 0 = SINGLE_CHANNEL 1 = UPTO_TWO_CHANNELS 2 = UPTO_FOUR_CHANNELS 3 = UPTO_EIGHT_CHANNELS 4 = UPTO_SIXTEEN_CHANNELS

Bit	Reset	Description
2:1	0x0	WAKE: Normally, the WAKE signal is automatically controller by the controller, It would be controller "low" or "high" via software to work around any bug (either in the controller or the device). 0 = AUTOMATICALLY_ASSERTS 1 = LOW 2 = HIGH
0	0x0	TRANSMISSION_MODE: TX mode of transmission. Refer to section 4.2 of the MIPI_HSI specification, v1.01. 0 = STREAM_MODE 1 = FRAME_MODE

34.5.2 MIPIHSI_GLOBAL_RX_CONTROL_0

HSI Global RX Control Register

Offset: 0x4 │ Read/Write: R/W │ Reset: 0xc8000000 (0b1100100xxxxxxxxxxxxxxx0x00000000)

Bit	Reset	Description
31	0x1	HSI_RX_SRC_RST: RX pad macro reset. Needed, so that rx pad macro can be reset in case of recovery without touching of controller reset. Please note that other side link also need to be reset at the same time when RX pad_macro is applied reset. 0 : Deassert the rx pad macro reset 1 : Assert the rx pad macro reset. Default is 1 so that pad macro reset will automatically be reset on power on reset.
30	0x1	HSI_FENCE_RD_ENABLE: Fence Read request enable - need to set this to avoid data coherency issues when writing to system memory. Keep it disabled (program 1'b0 to this bit) for successful operation in Tegra 4 series devices.
29:25	0x4	RX_CLK_TRIM_VALUE: Needed for RX pad macro.
9	0x0	RX_INTERNAL_WAKE: Needed for the devices that don't implement WAKE. In that case RX will have to assume that WAKE is always asserted. 0 : Look for external WAKE signal. 1 : External WAKE is not implemented. Assume WAKE is always there.
7:4	0x0	NUM_OF_RX_CHANNELS: Number of RX channel that software wants to use. The Channel descriptor field in the Frame will change accordingly. The total frame_length depends on this and RX reception mode. 0 = ONE_CHANNEL 1 = TWO_CHANNELS 2 = FOUR_CHANNELS 3 = EIGHT_CHANNELS 4 = SIXTEEN_CHANNELS
3:2	0x0	RECEIVER_DATA_FLOW_MODE: READY Timing depends on Receiver Data Flow Mode. Refer to section 4.1.2 of the MIPI_HSI specification, v1.01, for receiver data flow modes. Receiver_real_time_mode is not supported as its used only in case of error recovery. 0 = SYNCRONIZED_DATA_FLOW_MODE 1 = PIPELINED_DATA_FLOW_MODE 2 = RECEIVER_REAL_TIME_MODE
1	0x0	RX_MODE: RX Bit Reception mode. Please refer section 4.2 of MIPI_HSI spec v1.01. 0 = STREAM_MODE 1 = FRAME_MODE
0	0x0	RX_ENABLE: Enable/Disable RX. RX will work only if it is enabled. 0 = DISABLE 1 = ENABLE

34.5.3 MIPIHSI_PIO_DATA_PORT_0

HSI PIO Data/Word Register

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

Bit	Reset	Description
31:0	0x0	PIO_DATA: 32-bit data to be Transmitted through PIO. Only TX supports data transfers through PIO. The TX will start transmitting data as soon as software writes PIO word in this register.

34.5.4 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_0_0

HSI DMA TX Channel_0_Buffer_0 Control Register

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_0: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_0: priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_0: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting a new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_0: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.5 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_0_0

HSI DMA TX Channel_0_Buffer_1 Control Register

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_0: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_0: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting a new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_0: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.6 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_1_0

HSI DMA TX Channel_1_Buffer_0 Control Register

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_1: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_1: Priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_1: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting a new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_1: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.7 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_1_0

HSI DMA TX Channel_1_Buffer_1 Control Register

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000xxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_1: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_1: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_1: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.8 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_2_0

HSI DMA TX Channel_2_Buffer_0 Control Register

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_2: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.

Bit	Reset	Description
2	0x0	PRIORITY_TX_CH_2: Priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_2: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_2: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.9 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_2_0

HSI DMA TX Channel_2_Buffer_1 Control Register

Offset: 0x20 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_2: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_2: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_2: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.10 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_3_0

HSI DMA TX Channel_3_Buffer_0 Control Register

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_3: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_3: Priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_3: Stop fetching Data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE

Bit	Reset	Description
0	0x0	READY_BUF_0_TX_CH_3: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.11 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_3_0

HSI DMA TX Channel_3_Buffer_1 Control Register

Offset: 0x28 │ Read/Write: R/W │ Reset: 0x00000000 (0bx00000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_3: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_3: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_3: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.12 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_4_0

HSI DMA TX Channel_4_Buffer_0 Control Register

Offset: 0x2c │ Read/Write: R/W │ Reset: 0x00000000 (0bx00000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_4: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_4: Priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_4: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_4: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.13 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_4_0

HSI DMA TX Channel_4_Buffer_1 Control Register

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_4: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_4: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_4: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.14 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_5_0

HSI DMA TX Channel_5_Buffer_0 Control Register

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_5: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_5: priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round-robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_5: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_5: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.15 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_5_0

HSI DMA TX Channel_5_Buffer_1 Control Register

Offset: 0x38 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_5: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.

Bit	Reset	Description
1	0x0	HALT_DMA_BUF_1_TX_CH_5: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_5: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.16 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_6_0

HSI DMA TX Channel_6_Buffer_0 Control Register

Offset: 0x3c │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_6: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_6: priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_6: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_6: Ready/in-use semaphore that software sets when ready and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.17 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_6_0

HSI DMA TX Channel_6_Buffer_1 Control Register

Offset: 0x40 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_6: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_6: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_6: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.18 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_7_0

HSI DMA TX Channel_7_Buffer_0 Control Register

Offset: 0x44 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_7: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_7: priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_7: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_7: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.19 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_7_0

HSI DMA TX Channel_7_Buffer_1 Control Register

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_7: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_7: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_7: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.20 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_8_0

HSI DMA TX Channel_8_Buffer_0 Control Register

Offset: 0x4c | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_8: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.

Bit	Reset	Description
2	0x0	PRIORITY_TX_CH_8: priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_8: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_8: Ready/in-use semaphore that Software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.21 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_8_0

HSI DMA TX Channel_8_Buffer_1 Control Register

Offset: 0x50 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_8: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_8: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_8: Ready/in-use semaphore that Software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.22 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_9_0

HSI DMA TX Channel_9_Buffer_0 Control Register

Offset: 0x54 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_9: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_9: priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_9: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE

Bit	Reset	Description
0	0x0	READY_BUF_0_TX_CH_9: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.23 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_9_0

HSI DMA TX Channel_9_Buffer_1 Control Register

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_9: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_9: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_9: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.24 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_10_0

HSI DMA TX Channel_10_Buffer_0 Control Register

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_10: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_10: priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_10: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_10: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.25 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_10_0

HSI DMA TX Channel_10_Buffer_1 Control Register

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_10: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_10: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_10: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.26 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_11_0

HSI DMA TX Channel_11_Buffer_0 Control Register

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_11: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_11: priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_11: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_11: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.27 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_11_0

HSI DMA TX Channel_11_Buffer_1 Control Register

Offset: 0x68 | Read/Write: R/W | Reset: 0x00000000 (0bx00000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_11: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.

Bit	Reset	Description
1	0x0	HALT_DMA_BUF_1_TX_CH_11: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_11: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.28 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_12_0

HSI DMA TX Channel_12_Buffer_0 Control Register

Offset: 0x6c │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_12: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_12: priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_12: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_12: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.29 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_12_0

HSI DMA TX Channel_12_Buffer_1 Control Register

Offset: 0x70 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_12: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_12: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_12: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.30 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_13_0

HSI DMA TX Channel_13_Buffer_0 Control Register

Offset: 0x74 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_13: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_13: priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_13: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_13: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.31 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_13_0

HSI DMA TX Channel_13_Buffer_1 Control Register

Offset: 0x78 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_13: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_13: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_13: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.32 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_14_0

HSI DMA TX Channel_14_Buffer_0 Control Register

Offset: 0x7c | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_14: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.

Bit	Reset	Description
2	0x0	PRIORITY_TX_CH_14: priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_14: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_0_TX_CH_14: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.33 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_14_0

HSI DMA TX Channel_14_Buffer_1 Control Register

Offset: 0x80 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_14: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_14: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_14: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.34 MIPIHSI_DMA_CONTROL_TX_BUF_0_CH_15_0

HSI DMA TX Channel_15_Buffer_0 Control Register

Offset: 0x84 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxx000)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_TX_CH_15: Number of words to be transferred from Buffer_0. 0 = One 32-bit word. 0x7FFE = 127 KB.
2	0x0	PRIORITY_TX_CH_15: priority for this DMA channel. Affects both buffers of this channel. High priority channels are given preference over low priority. All the same priority channels follow the round robin scheme. 0 = HIGH 1 = LOW
1	0x0	HALT_DMA_BUF_0_TX_CH_15: Stop fetching data from BUF_0. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE

Bit	Reset	Description
0	0x0	READY_BUF_0_TX_CH_15: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.35 MIPIHSI_DMA_CONTROL_TX_BUF_1_CH_15_0

HSI DMA TX Channel_15_Buffer_1 Control Register

Offset: 0x88 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_TX_CH_15: Number of words to be transferred from Buffer_1. 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_BUF_1_TX_CH_15: Stop fetching data from BUF_1. Any data already fetched into the FIFO will still go out to the HSI Tx interface. 0 = DISABLE: Software needs to wait until hardware clears this bit before starting new transaction. 1 = ENABLE
0	0x0	READY_BUF_1_TX_CH_15: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.36 MIPIHSI_TX_CH_0_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_0 DMA Buffer_0 System Memory Address Register

Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_0: System Memory Address for DMA Buffer-0 for TX

34.5.37 MIPIHSI_TX_CH_0_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_0 DMA Buffer_1 System Memory Address Register

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_0: System Memory Address for DMA Buffer-1 for TX

34.5.38 MIPIHSI_TX_CH_1_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_1 DMA Buffer_0 System Memory Address Register

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_1: System Memory Address for DMA Buffer-0 for TX

34.5.39 MIPIHSI_TX_CH_1_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_1 DMA Buffer_1 System Memory Address Register

Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_1: System Memory Address for DMA Buffer-1 for TX

34.5.40 MIPIHSI_TX_CH_2_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_2 DMA Buffer_0 System Memory Address Register

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

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_2: System Memory Address for DMA Buffer-0 for TX

34.5.41 MIPIHSI_TX_CH_2_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_2 DMA Buffer_1 System Memory Address Register

Offset: 0xb4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_2: System Memory Address for DMA Buffer-1 for TX

34.5.42 MIPIHSI_TX_CH_3_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_3 DMA Buffer_0 System Memory Address Register

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

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_3: System Memory Address for DMA Buffer-0 for TX

34.5.43 MIPIHSI_TX_CH_3_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_3 DMA Buffer_1 System Memory Address Register

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_3: System Memory Address for DMA Buffer-1 for TX

34.5.44 MIPIHSI_TX_CH_4_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_4 DMA Buffer_0 System Memory Address Register

Offset: 0xcc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_4: System Memory Address for DMA Buffer-0 for TX

34.5.45 MIPIHSI_TX_CH_4_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_4 DMA Buffer_1 System Memory Address Register

Offset: 0xd4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_4: System Memory Address for DMA Buffer-1 for TX

34.5.46 MIPIHSI_TX_CH_5_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_5 DMA Buffer_0 System Memory Address Register

Offset: 0xdc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_5: System Memory Address for DMA Buffer-0 for TX

34.5.47 MIPIHSI_TX_CH_5_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_5 DMA Buffer_1 System Memory Address Register

Offset: 0xe4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_5: System Memory Address for DMA Buffer-1 for TX

34.5.48 MIPIHSI_TX_CH_6_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_6 DMA Buffer_0 System Memory Address Register

Offset: 0xec │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_6: System Memory Address for DMA Buffer-0 for TX

34.5.49 MIPIHSI_TX_CH_6_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_6 DMA Buffer_1 System Memory Address Register

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

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_6: System Memory Address for DMA Buffer-1 for TX

34.5.50 MIPIHSI_TX_CH_7_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_7 DMA Buffer_0 System Memory Address Register

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

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_7: System Memory Address for DMA Buffer-0 for TX

34.5.51 MIPIHSI_TX_CH_7_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_7 DMA Buffer_1 System Memory Address Register

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_7: System Memory Address for DMA Buffer-1 for TX

34.5.52 MIPIHSI_TX_CH_8_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_8 DMA Buffer_0 System Memory Address Register

Offset: 0x10c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_8: System Memory Address for DMA Buffer-0 for TX

34.5.53 MIPIHSI_TX_CH_8_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_8 DMA Buffer_1 System Memory Address Register

Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_8: System Memory Address for DMA Buffer-1 for TX

34.5.54 MIPIHSI_TX_CH_9_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_9 DMA Buffer_0 System Memory Address Register

Offset: 0x11c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_9: System Memory Address for DMA Buffer-0 for TX

34.5.55 MIPIHSI_TX_CH_9_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_9 DMA Buffer_1 System Memory Address Register

Offset: 0x124 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_9: System Memory Address for DMA Buffer-1 for TX

34.5.56 MIPIHSI_TX_CH_10_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_10 DMA Buffer_0 System Memory Address Register

Offset: 0x12c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_10: System Memory Address for DMA Buffer-0 for TX

34.5.57 MIPIHSI_TX_CH_10_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_10 DMA Buffer_1 System Memory Address Register

Offset: 0x134 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_10: System Memory Address for DMA Buffer-1 for TX

34.5.58 MIPIHSI_TX_CH_11_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_11 DMA Buffer_0 System Memory Address Register

Offset: 0x13c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_11: System Memory Address for DMA Buffer-0 for TX

34.5.59 MIPIHSI_TX_CH_11_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_11 DMA Buffer_1 System Memory Address Register

Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_11: System Memory Address for DMA Buffer-1 for TX

34.5.60 MIPIHSI_TX_CH_12_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_12 DMA Buffer_0 System Memory Address Register

Offset: 0x14c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_12: System Memory Address for DMA Buffer-0 for TX

34.5.61 MIPIHSI_TX_CH_12_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_12 DMA Buffer_1 System Memory Address Register

Offset: 0x154 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_12: System Memory Address for DMA Buffer-1 for TX

34.5.62 MIPIHSI_TX_CH_13_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_13 DMA Buffer_0 System Memory Address Register

Offset: 0x15c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_13: System Memory Address for DMA Buffer-0 for TX

34.5.63 MIPIHSI_TX_CH_13_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_13 DMA Buffer_1 System Memory Address Register

Offset: 0x164 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_13: System Memory Address for DMA Buffer-1 for TX

34.5.64 MIPIHSI_TX_CH_14_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_14 DMA Buffer_0 System Memory Address Register

Offset: 0x16c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_14: System Memory Address for DMA Buffer-0 for TX

34.5.65 MIPIHSI_TX_CH_14_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_14 DMA Buffer_1 System Memory Address Register

Offset: 0x174 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_14: System Memory Address for DMA Buffer-1 for TX

34.5.66 MIPIHSI_TX_CH_15_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_15 DMA Buffer_0 System Memory Address Register

Offset: 0x17c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_TX_CH_15: System Memory Address for DMA Buffer-0 for TX

34.5.67 MIPIHSI_TX_CH_15_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI TX Channel_15 DMA Buffer_1 System Memory Address Register

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_TX_CH_15: System Memory Address for DMA Buffer-1 for TX

34.5.68 MIPIHSI_TX_CH_0_DMA_BUFFER_STATUS_0

HSI TX Channel_0 DMA Buffer Status Register

Offset: 0x18c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_0: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_0: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.69 MIPIHSI_TX_CH_1_DMA_BUFFER_STATUS_0

HSI TX Channel_1 DMA Buffer Status Register

Offset: 0x190 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_1: Status indicating number of words received from the memory Buffer-1. Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_1: Status indicating number of words received from the memory Buffer-0. Ranges from 0 to 128 KB (0x8000 words).

34.5.70 MIPIHSI_TX_CH_2_DMA_BUFFER_STATUS_0

HSI TX Channel_2 DMA Buffer Status Register

Offset: 0x194 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_2: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_2: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.71 MIPIHSI_TX_CH_3_DMA_BUFFER_STATUS_0

HSI TX Channel_3 DMA Buffer Status Register

Offset: 0x198 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_3: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_3: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.72 MIPIHSI_TX_CH_4_DMA_BUFFER_STATUS_0

HSI TX Channel_4 DMA Buffer Status Register

Offset: 0x19c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_4: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_4: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.73 MIPIHSI_TX_CH_5_DMA_BUFFER_STATUS_0

HSI TX Channel_5 DMA Buffer Status Register

Offset: 0x1a0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_5: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_5: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.74 MIPIHSI_TX_CH_6_DMA_BUFFER_STATUS_0

HSI TX Channel_6 DMA Buffer Status Register

Offset: 0x1a4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_6: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_6: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.75 MIPIHSI_TX_CH_7_DMA_BUFFER_STATUS_0

HSI TX Channel_7 DMA Buffer Status Register

Offset: 0x1a8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_7: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128B (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_7: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.76 MIPIHSI_TX_CH_8_DMA_BUFFER_STATUS_0

HSI TX Channel_8 DMA Buffer Status Register

Offset: 0x1ac | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_8: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_8: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.77 MIPIHSI_TX_CH_9_DMA_BUFFER_STATUS_0

HSI TX Channel_9 DMA Buffer Status Register

Offset: 0x1b0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_9: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_9: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.78 MIPIHSI_TX_CH_10_DMA_BUFFER_STATUS_0

HSI TX Channel_10 DMA Buffer Status Register

Offset: 0x1b4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_10: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_10: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.79 MIPIHSI_TX_CH_11_DMA_BUFFER_STATUS_0

HSI TX Channel_11 DMA Buffer Status Register

Offset: 0x1b8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_11: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_11: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.80 MIPIHSI_TX_CH_12_DMA_BUFFER_STATUS_0

HSI TX Channel_12 DMA Buffer Status Register

Offset: 0x1bc | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_12: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_12: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.81 MIPIHSI_TX_CH_13_DMA_BUFFER_STATUS_0

HSI TX Channel_13 DMA Buffer Status Register

Offset: 0x1c0 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_13: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_13: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.82 MIPIHSI_TX_CH_14_DMA_BUFFER_STATUS_0

HSI TX Channel_14 DMA Buffer Status Register

Offset: 0x1c4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_14: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_14: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.83 MIPIHSI_TX_CH_15_DMA_BUFFER_STATUS_0

HSI TX Channel_15 DMA Buffer Status Register

Offset: 0x1c8 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_TX_CH_15: Status indicating number of words received from the memory Buffer-1.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_TX_CH_15: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.84 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_0_0

HSI DMA RX Channel_0_Buffer_0 Control Register

Offset: 0x1cc | Read/Write: R/W | Reset: 0x00000000 (0bx0000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_0: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_0: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting new transaction.
0	0x0	READY_BUF_0_RX_CH_0: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.85 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_0_0

HSI DMA RX Channel_0_Buffer_1 Control Register

Offset: 0x1d0 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_0: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_0: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.86 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_1_0

HSI DMA RX Channel_1_Buffer_0 Control Register

Offset: 0x1d4 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_1: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_1: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting new transaction.
0	0x0	READY_BUF_0_RX_CH_1: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.87 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_1_0

HSI DMA RX Channel_1_Buffer_1 Control Register

Offset: 0x1d8 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_1: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_1: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.88 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_2_0

HSI DMA RX Channel_2_Buffer_0 Control Register

Offset: 0x1dc │ Read/Write: R/W │ Reset: 0x00000000 (0bx0000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_2: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_2: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting new transaction.
0	0x0	READY_BUF_0_RX_CH_2: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.89 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_2_0

HSI DMA RX Channel_2_Buffer_1 Control Register

Offset: 0x1e0 │ Read/Write: R/W │ Reset: 0x00000000 (0bx0000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_2: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_2: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.90 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_3_0

HSI DMA RX Channel_3_Buffer_0 Control Register

Offset: 0x1e4 │ Read/Write: R/W │ Reset: 0x00000000 (0bx0000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_3: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_3: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting new transaction.
0	0x0	READY_BUF_0_RX_CH_3: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.91 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_3_0

HSI DMA RX Channel_3_Buffer_1 Control Register

Offset: 0x1e8 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_3: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_3: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.92 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_4_0

HSI DMA RX Channel_4_Buffer_0 Control Register

Offset: 0x1ec | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_4: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_4: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting new transaction.
0	0x0	READY_BUF_0_RX_CH_4: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.93 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_4_0

HSI DMA RX Channel_4_Buffer_1 Control Register

Offset: 0x1f0 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_4: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_4: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.94 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_5_0

HSI DMA RX Channel_5_Buffer_0 Control Register

Offset: 0x1f4 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_5: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_5: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting new transaction.
0	0x0	READY_BUF_0_RX_CH_5: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.95 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_5_0

HSI DMA RX Channel_5_Buffer_1 Control Register

Offset: 0x1f8 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_5: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_5: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.96 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_6_0

HSI DMA RX Channel_6_Buffer_0 Control Register

Offset: 0x1fc | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_6: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_6: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting new transaction.
0	0x0	READY_BUF_0_RX_CH_6: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.97 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_6_0

HSI DMA RX Channel_6_Buffer_1 Control Register

Offset: 0x200 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_6: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_6: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.98 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_7_0

HSI DMA RX Channel_7_Buffer_0 Control Register

Offset: 0x204 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_7: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_7: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting a new transaction.
0	0x0	READY_BUF_0_RX_CH_7: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.99 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_7_0

HSI DMA RX Channel_7_Buffer_1 Control Register

Offset: 0x208 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_7: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_7: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.100 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_8_0

HSI DMA RX Channel_8_Buffer_0 Control Register

Offset: 0x20c │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_8: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_8: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting a new transaction.
0	0x0	READY_BUF_0_RX_CH_8: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.101 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_8_0

HSI DMA RX Channel_8_Buffer_1 Control Register

Offset: 0x210 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_8: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_8: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.102 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_9_0

HSI DMA RX Channel_9_Buffer_0 Control Register

Offset: 0x214 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_9: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_9: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting a new transaction.
0	0x0	READY_BUF_0_RX_CH_9: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.103 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_9_0

HSI DMA RX Channel_9_Buffer_1 Control Register

Offset: 0x218 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_9: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_9: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.104 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_10_0

HSI DMA RX Channel_10_Buffer_0 Control Register

Offset: 0x21c | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_10: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_10: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting a new transaction.
0	0x0	READY_BUF_0_RX_CH_10: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.105 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_10_0

HSI DMA RX Channel_10_Buffer_1 Control Register

Offset: 0x220 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_10: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_10: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.106 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_11_0

HSI DMA RX Channel_11_Buffer_0 Control Register

Offset: 0x224 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_11: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_11: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting a new transaction.
0	0x0	READY_BUF_0_RX_CH_11: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.107 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_11_0

HSI DMA RX Channel_11_Buffer_1 Control Register

Offset: 0x228 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_11: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_11: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.108 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_12_0

HSI DMA RX Channel_12_Buffer_0 Control Register

Offset: 0x22c │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_12: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_12: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting a new transaction.
0	0x0	READY_BUF_0_RX_CH_12: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.109 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_12_0

HSI DMA RX Channel_12_Buffer_1 Control Register

Offset: 0x230 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_12: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_12: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.110 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_13_0

HSI DMA RX Channel_13_Buffer_0 Control Register

Offset: 0x234 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_13: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_13: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting a new transaction.
0	0x0	READY_BUF_0_RX_CH_13: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.111 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_13_0

HSI DMA RX Channel_13_Buffer_1 Control Register

Offset: 0x238 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_13: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_13: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.112 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_14_0

HSI DMA RX Channel_14_Buffer_0 Control Register

Offset: 0x23c │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_14: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_14: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting a new transaction.
0	0x0	READY_BUF_0_RX_CH_14: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.113 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_14_0

HSI DMA RX Channel_14_Buffer_1 Control Register

Offset: 0x240 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_14: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_14: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.114 MIPIHSI_DMA_CONTROL_RX_BUF_0_CH_15_0

HSI DMA RX Channel_15_Buffer_0 Control Register

Offset: 0x244 │ Read/Write: R/W │ Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxx00)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_0_RX_CH_15: Number of words to be transferred from Buffer_0 0 = One 32-bit word. 0x7FFE = 127 KB.
1	0x0	HALT_DMA_RX_CH_15: Stop writing data to BUF_0. Hardware will clear the READY_BUF bit on HALT. Software needs to wait until hardware clears this bit before starting a new transaction.
0	0x0	READY_BUF_0_RX_CH_15: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.115 MIPIHSI_DMA_CONTROL_RX_BUF_1_CH_15_0

HSI DMA RX Channel_15_Buffer_1 Control Register

Offset: 0x248 | Read/Write: R/W | Reset: 0x00000000 (0bx000000000000000xxxxxxxxxxxxxxx0)

Bit	Reset	Description
30:16	0x0	LENGTH_DMA_BUF_1_RX_CH_15: Number of words to be transferred from Buffer_1 0 = One 32-bit word. 0x7FFE = 127 KB.
0	0x0	READY_BUF_1_RX_CH_15: Ready/in-use semaphore that software sets when ready, and hardware clears it when no longer in use. 0 = NO_LONGER_IN_USE 1 = IN_USE

34.5.116 MIPIHSI_RX_CH_0_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_0 DMA Buffer_0 System Memory Address Register

Offset: 0x24c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_0: System Memory Address for DMA Buffer-0 for RX

34.5.117 MIPIHSI_RX_CH_0_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_0 DMA Buffer_1 System Memory Address Register

Offset: 0x254 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_0: System Memory Address for DMA Buffer-1 for RX

34.5.118 MIPIHSI_RX_CH_1_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_1 DMA Buffer_0 System Memory Address Register

Offset: 0x25c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_1: System Memory Address for DMA Buffer-0 for RX

34.5.119 MIPIHSI_RX_CH_1_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_1 DMA Buffer_1 System Memory Address Register

Offset: 0x264 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_1: System Memory Address for DMA Buffer-1 for RX

34.5.120 MIPIHSI_RX_CH_2_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_2 DMA Buffer_0 System Memory Address Register

Offset: 0x26c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_2: System Memory Address for DMA Buffer-0 for RX

34.5.121 MIPIHSI_RX_CH_2_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_2 DMA Buffer_1 System Memory Address Register

Offset: 0x274 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_2: System Memory Address for DMA Buffer-1 for RX

34.5.122 MIPIHSI_RX_CH_3_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_3 DMA Buffer_0 System Memory Address Register

Offset: 0x27c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_3: System Memory Address for DMA Buffer-0 for RX

34.5.123 MIPIHSI_RX_CH_3_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_3 DMA Buffer_1 System Memory Address Register

Offset: 0x284 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_3: System Memory Address for DMA Buffer-1 for RX

34.5.124 MIPIHSI_RX_CH_4_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_4 DMA Buffer_0 System Memory Address Register

Offset: 0x28c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_4: System Memory Address for DMA Buffer-0 for RX

34.5.125 MIPIHSI_RX_CH_4_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_4 DMA Buffer_1 System Memory Address Register

Offset: 0x294 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_4: System Memory Address for DMA Buffer-1 for RX

34.5.126 MIPIHSI_RX_CH_5_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_5 DMA Buffer_0 System Memory Address Register

Offset: 0x29c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_5: System Memory Address for DMA Buffer-0 for RX

34.5.127 MIPIHSI_RX_CH_5_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_5 DMA Buffer_1 System Memory Address Register

Offset: 0x2a4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_5: System Memory Address for DMA Buffer-1 for RX

34.5.128 MIPIHSI_RX_CH_6_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_6 DMA Buffer_0 System Memory Address Register

Offset: 0x2ac │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_6: System Memory Address for DMA Buffer-0 for RX

34.5.129 MIPIHSI_RX_CH_6_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_6 DMA Buffer_1 System Memory Address Register

Offset: 0x2b4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_6: System Memory Address for DMA Buffer-1 for RX

34.5.130 MIPIHSI_RX_CH_7_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_7 DMA Buffer_0 System Memory Address Register

Offset: 0x2bc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_7: System Memory Address for DMA Buffer-0 for RX

34.5.131 MIPIHSI_RX_CH_7_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_7 DMA Buffer_1 System Memory Address Register

Offset: 0x2c4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_7: System Memory Address for DMA Buffer-1 for RX

34.5.132 MIPIHSI_RX_CH_8_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_8 DMA Buffer_0 System Memory Address Register

Offset: 0x2cc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_8: System Memory Address for DMA Buffer-0 for RX

34.5.133 MIPIHSI_RX_CH_8_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_8 DMA Buffer_1 System Memory Address Register

Offset: 0x2d4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_8: System Memory Address for DMA Buffer-1 for RX

34.5.134 MIPIHSI_RX_CH_9_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_9 DMA Buffer_0 System Memory Address Register

Offset: 0x2dc │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_9: System Memory Address for DMA Buffer-0 for RX

34.5.135 MIPIHSI_RX_CH_9_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_9 DMA Buffer_1 System Memory Address Register

Offset: 0x2e4 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_9: System Memory Address for DMA Buffer-1 for RX

34.5.136 MIPIHSI_RX_CH_10_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_10 DMA Buffer_0 System Memory Address Register

Offset: 0x2ec │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_10: System Memory Address for DMA Buffer-0 for RX

34.5.137 MIPIHSI_RX_CH_10_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_10 DMA Buffer_1 System Memory Address Register

Offset: 0x2f4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_10: System Memory Address for DMA Buffer-1 for RX

34.5.138 MIPIHSI_RX_CH_11_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_11 DMA Buffer_0 System Memory Address Register

Offset: 0x2fc | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_11: System Memory Address for DMA Buffer-0 for RX

34.5.139 MIPIHSI_RX_CH_11_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_11 DMA Buffer_1 System Memory Address Register

Offset: 0x304 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_11: System Memory Address for DMA Buffer-1 for RX

34.5.140 MIPIHSI_RX_CH_12_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_12 DMA Buffer_0 System Memory Address Register

Offset: 0x30c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_12: System Memory Address for DMA Buffer-0 for RX

34.5.141 MIPIHSI_RX_CH_12_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_12 DMA Buffer_1 System Memory Address Register

Offset: 0x314 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_12: System Memory Address for DMA Buffer-1 for RX

34.5.142 MIPIHSI_RX_CH_13_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_13 DMA Buffer_0 System Memory Address Register

Offset: 0x31c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_13: System Memory Address for DMA Buffer-0 for RX

34.5.143 MIPIHSI_RX_CH_13_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_13 DMA Buffer_1 System Memory Address Register

Offset: 0x324 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_13: System Memory Address for DMA Buffer-1 for RX

34.5.144 MIPIHSI_RX_CH_14_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_14 DMA Buffer_0 System Memory Address Register

Offset: 0x32c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_14: System Memory Address for DMA Buffer-0 for RX

34.5.145 MIPIHSI_RX_CH_14_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_14 DMA Buffer_1 System Memory Address Register

Offset: 0x334 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_14: System Memory Address for DMA Buffer-1 for RX

34.5.146 MIPIHSI_RX_CH_15_DMA_BUFFER_0_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_15 DMA Buffer_0 System Memory Address Register

Offset: 0x33c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_0_RX_CH_15: System Memory Address for DMA Buffer-0 for RX

34.5.147 MIPIHSI_RX_CH_15_DMA_BUFFER_1_SYSTEM_MEMORY_ADDR_0

HSI RX Channel_15 DMA Buffer_1 System Memory Address Register

Offset: 0x344 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SYSTEM_MEM_ADDR_BUF_1_RX_CH_15: System Memory Address for DMA Buffer-1 for RX

34.5.148 MIPIHSI_RX_CH_0_DMA_BUFFER_STATUS_0

HSI RX Channel_0 DMA Buffer Status Register

Offset: 0x34c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_0: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB(0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_0: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB(0x8000 words).

34.5.149 MIPIHSI_RX_CH_1_DMA_BUFFER_STATUS_0

HSI RX Channel_1 DMA Buffer Status Register

Offset: 0x350 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_1: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB(0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_1: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB(0x8000 words).

34.5.150 MIPIHSI_RX_CH_2_DMA_BUFFER_STATUS_0

HSI RX Channel_2 DMA Buffer Status Register

Offset: 0x354 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_2: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_2: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.151 MIPIHSI_RX_CH_3_DMA_BUFFER_STATUS_0

HSI RX Channel_3 DMA Buffer Status Register

Offset: 0x358 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_3: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_3: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.152 MIPIHSI_RX_CH_4_DMA_BUFFER_STATUS_0

HSI RX Channel_4 DMA Buffer Status Register

Offset: 0x35c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_4: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_4: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.153 MIPIHSI_RX_CH_5_DMA_BUFFER_STATUS_0

HSI RX Channel_5 DMA Buffer Status Register

Offset: 0x360 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_5: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_5: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.154 MIPIHSI_RX_CH_6_DMA_BUFFER_STATUS_0

HSI RX Channel_6 DMA Buffer Status Register

Offset: 0x364 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_6: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_6: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.155 MIPIHSI_RX_CH_7_DMA_BUFFER_STATUS_0

HSI RX Channel_7 DMA Buffer Status Register

Offset: 0x368 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_7: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_7: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.156 MIPIHSI_RX_CH_8_DMA_BUFFER_STATUS_0

HSI RX Channel_8 DMA Buffer Status Register

Offset: 0x36c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_8: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_8: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.157 MIPIHSI_RX_CH_9_DMA_BUFFER_STATUS_0

HSI RX Channel_9 DMA Buffer Status Register

Offset: 0x370 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_9: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_9: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.158 MIPIHSI_RX_CH_10_DMA_BUFFER_STATUS_0

HSI RX Channel_10 DMA Buffer Status Register

Offset: 0x374 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_10: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_10: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.159 MIPIHSI_RX_CH_11_DMA_BUFFER_STATUS_0

HSI RX Channel_11 DMA Buffer Status Register

Offset: 0x378 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_11: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_11: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.160 MIPIHSI_RX_CH_12_DMA_BUFFER_STATUS_0

HSI RX Channel_12 DMA Buffer Status Register

Offset: 0x37c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_12: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_12: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.161 MIPIHSI_RX_CH_13_DMA_BUFFER_STATUS_0

HSI RX Channel_13 DMA Buffer Status Register

Offset: 0x380 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_13: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_13: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.162 MIPIHSI_RX_CH_14_DMA_BUFFER_STATUS_0

HSI RX Channel_14 DMA Buffer Status Register

Offset: 0x384 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_14: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_14: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.163 MIPIHSI_RX_CH_15_DMA_BUFFER_STATUS_0

HSI RX Channel_15 DMA Buffer Status Register

Offset: 0x388 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	WORDS_READ_STATUS_FOR_BUFFER_1_RX_CH_15: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).
15:0	X	WORDS_READ_STATUS_FOR_BUFFER_0_RX_CH_15: Status indicating number of words received from the memory Buffer-0.Ranges from 0 to 128 KB (0x8000 words).

34.5.164 MIPIHSI_TX_DMA_BUFFER_COMPLETE_INTERRUPT_ENABLE_0

HSI TX DMA Buffer Complete Interrupt Enable Register

Offset: 0x38c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_15: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
30	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_15: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
29	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_14: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
28	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_14: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
27	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_13: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
26	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_13: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
25	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_12: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
24	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_12: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
23	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_11: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_11: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
21	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_10: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
20	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_10: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
19	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_9: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
18	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_9: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
17	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_8: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
16	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_8: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
15	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_7: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
14	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_7: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
13	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_6: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
12	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_6: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
11	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_5: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
10	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_5: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
9	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_4: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_4: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
7	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_3: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
6	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_3: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
5	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_2: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
4	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_2: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
3	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_1: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
2	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_1: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
1	0x0	BUFFER_1_COMPLETE_INTR_EN_TX_CH_0: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
0	0x0	BUFFER_0_COMPLETE_INTR_EN_TX_CH_0: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE

34.5.165 MIPIHSI_TX_DMA_BUFFER_COMPLETE_INTERRUPT_STATUS_0

HSI TX DMA Buffer Complete Interrupt Status Register

Offset: 0x390 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_15: 0 = NO_INTR 1 = INTR
30	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_15: 0 = NO_INTR 1 = INTR
29	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_14: 0 = NO_INTR 1 = INTR
28	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_14: 0 = NO_INTR 1 = INTR

Bit	Reset	Description
27	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_13: 0 = NO_INTR 1 = INTR
26	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_13: 0 = NO_INTR 1 = INTR
25	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_12: 0 = NO_INTR 1 = INTR
24	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_12: 0 = NO_INTR 1 = INTR
23	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_11: 0 = NO_INTR 1 = INTR
22	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_11: 0 = NO_INTR 1 = INTR
21	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_10: 0 = NO_INTR 1 = INTR
20	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_10: 0 = NO_INTR 1 = INTR
19	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_9: 0 = NO_INTR 1 = INTR
18	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_9: 0 = NO_INTR 1 = INTR
17	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_8: 0 = NO_INTR 1 = INTR
16	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_8: 0 = NO_INTR 1 = INTR
15	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_7: 0 = NO_INTR 1 = INTR
14	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_7: 0 = NO_INTR 1 = INTR
13	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_6: 0 = NO_INTR 1 = INTR
12	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_6: 0 = NO_INTR 1 = INTR
11	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_5: 0 = NO_INTR 1 = INTR
10	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_5: 0 = NO_INTR 1 = INTR

Bit	Reset	Description
9	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_4: 0 = NO_INTR 1 = INTR
8	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_4: 0 = NO_INTR 1 = INTR
7	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_3: 0 = NO_INTR 1 = INTR
6	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_3: 0 = NO_INTR 1 = INTR
5	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_2: 0 = NO_INTR 1 = INTR
4	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_2: 0 = NO_INTR 1 = INTR
3	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_1: 0 = NO_INTR 1 = INTR
2	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_1: 0 = NO_INTR 1 = INTR
1	0x0	BUFFER_1_COMPLETE_INTR_STATUS_TX_CH_0: 0 = NO_INTR 1 = INTR
0	0x0	BUFFER_0_COMPLETE_INTR_STATUS_TX_CH_0: 0 = NO_INTR 1 = INTR

34.5.166 MIPIHSI_RX_DMA_BUFFER_COMPLETE_INTERRUPT_ENABLE_0

HSI RX DMA Buffer Complete Interrupt Enable Register

Offset: 0x394 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_15: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
30	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_15: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
29	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_14: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
28	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_14: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_13: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
26	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_13: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
25	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_12: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
24	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_12: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
23	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_11: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
22	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_11: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
21	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_10: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
20	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_10: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
19	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_9: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
18	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_9: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
17	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_8: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
16	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_8: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
15	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_7: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
14	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_7: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
13	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_6: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
12	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_6: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
11	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_5: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
10	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_5: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
9	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_4: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
8	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_4: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
7	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_3: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
6	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_3: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
5	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_2: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
4	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_2: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
3	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_1: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
2	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_1: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
1	0x0	BUFFER_1_COMPLETE_INTR_EN_RX_CH_0: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE
0	0x0	BUFFER_0_COMPLETE_INTR_EN_RX_CH_0: Interrupt Enable bit for corresponding Interrupt Status bit. 0 = DISABLE 1 = ENABLE

34.5.167 MIPIHSI_RX_DMA_BUFFER_COMPLETE_INTERRUPT_STATUS_0

HSI RX DMA Buffer Complete Interrupt Status Register

Offset: 0x398 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_15: 0 = NO_INTR 1 = INTR
30	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_15: 0 = NO_INTR 1 = INTR
29	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_14: 0 = NO_INTR 1 = INTR
28	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_14: 0 = NO_INTR 1 = INTR
27	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_13: 0 = NO_INTR 1 = INTR
26	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_13: 0 = NO_INTR 1 = INTR
25	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_12: 0 = NO_INTR 1 = INTR
24	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_12: 0 = NO_INTR 1 = INTR
23	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_11: 0 = NO_INTR 1 = INTR
22	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_11: 0 = NO_INTR 1 = INTR
21	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_10: 0 = NO_INTR 1 = INTR
20	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_10: 0 = NO_INTR 1 = INTR
19	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_9: 0 = NO_INTR 1 = INTR
18	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_9: 0 = NO_INTR 1 = INTR
17	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_8: 0 = NO_INTR 1 = INTR
16	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_8: 0 = NO_INTR 1 = INTR

Bit	Reset	Description
15	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_7: 0 = NO_INTR 1 = INTR
14	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_7: 0 = NO_INTR 1 = INTR
13	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_6: 0 = NO_INTR 1 = INTR
12	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_6: 0 = NO_INTR 1 = INTR
11	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_5: 0 = NO_INTR 1 = INTR
10	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_5: 0 = NO_INTR 1 = INTR
9	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_4: 0 = NO_INTR 1 = INTR
8	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_4: 0 = NO_INTR 1 = INTR
7	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_3: 0 = NO_INTR 1 = INTR
6	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_3: 0 = NO_INTR 1 = INTR
5	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_2: 0 = NO_INTR 1 = INTR
4	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_2: 0 = NO_INTR 1 = INTR
3	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_1: 0 = NO_INTR 1 = INTR
2	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_1: 0 = NO_INTR 1 = INTR
1	0x0	BUFFER_1_COMPLETE_INTR_STATUS_RX_CH_0: 0 = NO_INTR 1 = INTR
0	0x0	BUFFER_0_COMPLETE_INTR_STATUS_RX_CH_0: 0 = NO_INTR 1 = INTR

34.5.168 MIPIHSI_RX_ERROR_INTERRUPT_ENABLE_0

HSI RX Error Interrupt Enable Register

Offset: 0x39c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_15_EN: 0 = DISABLE 1 = ENABLE
14	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_14_EN: 0 = DISABLE 1 = ENABLE
13	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_13_EN: 0 = DISABLE 1 = ENABLE
12	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_12_EN: 0 = DISABLE 1 = ENABLE
11	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_11_EN: 0 = DISABLE 1 = ENABLE
10	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_10_EN: 0 = DISABLE 1 = ENABLE
9	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_9_EN: 0 = DISABLE 1 = ENABLE
8	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_8_EN: 0 = DISABLE 1 = ENABLE
7	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_7_EN: 0 = DISABLE 1 = ENABLE
6	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_6_EN: 0 = DISABLE 1 = ENABLE
5	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_5_EN: 0 = DISABLE 1 = ENABLE
4	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_4_EN: 0 = DISABLE 1 = ENABLE
3	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_3_EN: 0 = DISABLE 1 = ENABLE
2	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_2_EN: 0 = DISABLE 1 = ENABLE
1	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_1_EN: 0 = DISABLE 1 = ENABLE
0	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_0_EN: 0 = DISABLE 1 = ENABLE

34.5.169 MIPIHSI_RX_ERROR_INTERRUPT_STATUS_0

HSI RX Error Interrupt Status Register

Offset: 0x3a0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_15: 0 = NO_INTR 1 = INTR
14	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_14: 0 = NO_INTR 1 = INTR
13	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_13: 0 = NO_INTR 1 = INTR
12	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_12: 0 = NO_INTR 1 = INTR
11	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_11: 0 = NO_INTR 1 = INTR
10	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_10: 0 = NO_INTR 1 = INTR
9	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_9: 0 = NO_INTR 1 = INTR
8	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_8: 0 = NO_INTR 1 = INTR
7	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_7: 0 = NO_INTR 1 = INTR
6	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_6: 0 = NO_INTR 1 = INTR
5	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_5: 0 = NO_INTR 1 = INTR
4	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_4: 0 = NO_INTR 1 = INTR
3	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_3: 0 = NO_INTR 1 = INTR
2	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_2: 0 = NO_INTR 1 = INTR
1	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_1: 0 = NO_INTR 1 = INTR
0	0x0	RX_FRAME_RECEIVED_WITHOUT_DMA_CH_0: 0 = NO_INTR 1 = INTR

34.5.170 MIPIHSI_RX_ERROR_DATA_FRAME_0

HSI RX Error Data Frame Register

Offset: 0x3a4 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RX_ERROR_DATA: Most recently received Data for a channel without a DMA setup.

34.5.171 MIPIHSI_RX_NEW_DATA_WR_COMP_INTERRUPT_ENABLE_0

HSI RX New Data Write Completion Interrupt Enable Register

This interrupt will be asserted when controller is done writing the current burst onto the AHB. Software may want to disable this interrupt in case of not being informed on every burst.

Offset: 0x3a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15	0x0	RX_NEW_DATA_WR_COMP_CH_15_EN: 0 = DISABLE 1 = ENABLE
14	0x0	RX_NEW_DATA_WR_COMP_CH_14_EN: 0 = DISABLE 1 = ENABLE
13	0x0	RX_NEW_DATA_WR_COMP_CH_13_EN: 0 = DISABLE 1 = ENABLE
12	0x0	RX_NEW_DATA_WR_COMP_CH_12_EN: 0 = DISABLE 1 = ENABLE
11	0x0	RX_NEW_DATA_WR_COMP_CH_11_EN: 0 = DISABLE 1 = ENABLE
10	0x0	RX_NEW_DATA_WR_COMP_CH_10_EN: 0 = DISABLE 1 = ENABLE
9	0x0	RX_NEW_DATA_WR_COMP_CH_9_EN: 0 = DISABLE 1 = ENABLE
8	0x0	RX_NEW_DATA_WR_COMP_CH_8_EN: 0 = DISABLE 1 = ENABLE
7	0x0	RX_NEW_DATA_WR_COMP_CH_7_EN: 0 = DISABLE 1 = ENABLE
6	0x0	RX_NEW_DATA_WR_COMP_CH_6_EN: 0 = DISABLE 1 = ENABLE
5	0x0	RX_NEW_DATA_WR_COMP_CH_5_EN: 0 = DISABLE 1 = ENABLE
4	0x0	RX_NEW_DATA_WR_COMP_CH_4_EN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x0	RX_NEW_DATA_WR_COMP_CH_3_EN: 0 = DISABLE 1 = ENABLE
2	0x0	RX_NEW_DATA_WR_COMP_CH_2_EN: 0 = DISABLE 1 = ENABLE
1	0x0	RX_NEW_DATA_WR_COMP_CH_1_EN: 0 = DISABLE 1 = ENABLE
0	0x0	RX_NEW_DATA_WR_COMP_CH_0_EN: 0 = DISABLE 1 = ENABLE

34.5.172 MIPIHSI_RX_NEW_DATA_WR_COMP_INTERRUPT_STATUS_0

HSI RX New Data Write Completion Interrupt Status Register

Offset: 0x3ac │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15	0x0	RX_NEW_DATA_WR_COMP_CH_15_INTR: 0 = NO_INTR 1 = INTR
14	0x0	RX_NEW_DATA_WR_COMP_CH_14_INTR: 0 = NO_INTR 1 = INTR
13	0x0	RX_NEW_DATA_WR_COMP_CH_13_INTR: 0 = NO_INTR 1 = INTR
12	0x0	RX_NEW_DATA_WR_COMP_CH_12_INTR: 0 = NO_INTR 1 = INTR
11	0x0	RX_NEW_DATA_WR_COMP_CH_11_INTR: 0 = NO_INTR 1 = INTR
10	0x0	RX_NEW_DATA_WR_COMP_CH_10_INTR: 0 = NO_INTR 1 = INTR
9	0x0	RX_NEW_DATA_WR_COMP_CH_9_INTR: 0 = NO_INTR 1 = INTR
8	0x0	RX_NEW_DATA_WR_COMP_CH_8_INTR: 0 = NO_INTR 1 = INTR
7	0x0	RX_NEW_DATA_WR_COMP_CH_7_INTR: 0 = NO_INTR 1 = INTR
6	0x0	RX_NEW_DATA_WR_COMP_CH_6_INTR: 0 = NO_INTR 1 = INTR
5	0x0	RX_NEW_DATA_WR_COMP_CH_5_INTR: 0 = NO_INTR 1 = INTR

Bit	Reset	Description
4	0x0	RX_NEW_DATA_WR_COMP_CH_4_INTR: 0 = NO_INTR 1 = INTR
3	0x0	RX_NEW_DATA_WR_COMP_CH_3_INTR: 0 = NO_INTR 1 = INTR
2	0x0	RX_NEW_DATA_WR_COMP_CH_2_INTR: 0 = NO_INTR 1 = INTR
1	0x0	RX_NEW_DATA_WR_COMP_CH_1_INTR: 0 = NO_INTR 1 = INTR
0	0x0	RX_NEW_DATA_WR_COMP_CH_0_INTR: 0 = NO_INTR 1 = INTR

34.5.173 MIPIHSI_MISC_1_CONTROL_0

HSI Misc 1 Control Register

Offset: 0x3b0 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxx000x000xxx0xxxx00000000)

Bit	R/W	Reset	Description
31	RO	X	TX_WAKE_STATE: TX WAKE state This signal is a real-time sample. It is not synchronized, and should only be used for debug.
30	RO	X	TX_FLAG_STATE: TX FLAG state This signal is a real-time sample. It is not synchronized, and should only be used for debug.
29	RO	X	TX_DATA_STATE: TX DATA state This signal is a real-time sample. It is not synchronized, and should only be used for debug.
28	RO	X	TX_READY_STATE: TX READY state This signal is a real-time sample. It is not synchronized, and should only be used for debug.
27	RO	X	RX_WAKE_STATE: RX WAKE state This signal is a real-time sample. It is not synchronized, and should only be used for debug.
26	RO	X	RX_FLAG_STATE: RX FLAG state This signal is a real-time sample. It is not synchronized, and should only be used for debug.
25	RO	X	RX_DATA_STATE: RX DATA state This signal is a real-time sample. It is not synchronized, and should only be used for debug.
24	RO	X	RX_READY_STATE: RX READY state This signal is a real-time sample. It is not synchronized, and should only be used for debug.
22	RW	0x0	TX_PIO_INT_STS: Transmitter Programmed IO Complete Interrupt Status. This is asserted when controller is done sending the PIO frame out on HSI interface. Now software can initiate the next PIO frame. 0 = NO_INTR 1 = INTR
21	RW	0x0	RX_BREAK_FRAME_INT_STS: Break Frame Received Interrupt Status. frame i.e., 33 to 38 zeros in a row). In other words, when the Frame Start bit is received as a zero instead of a 1. 0 = NO_INTR : Break frame is when, during Frame Transmission Mode, more consecutive zeros have been received than possible in a 1 = INTR
20	RW	0x0	RX_FRAME_TO_INT_STS: Receiver Frame Timeout Interrupt Status. This is asserted when timeout counter expires. 0 = NO_INTR 1 = INTR

Bit	R/W	Reset	Description
18	RW	0x0	TX_PIO_INT_EN: Transmitter Programmed IO Complete Interrupt Enable. 0 = DISABLE 1 = ENABLE
17	RW	0x0	RX_BREAK_FRAME_INT_EN: Break Frame Received Interrupt Enable. 0 = DISABLE 1 = ENABLE
16	RW	0x0	RX_FRAME_TO_INT_EN: Receiver Frame Timeout Interrupt Enable. 0 = DISABLE 1 = ENABLE
12	RW	0x0	MIPIHSI_CLK_OVERRIDE: clk override bit used for SLCG. Software should set this bit as 0 normally to let the SLCG happen. 0 : SLCG enable. 1 : No SLCG.
7:0	RW	0x0	MAX_RX_BURST_COUNT: Receiver Frame Burst Counter. Has effect only during Pipelined Data Flow. Note that values 0 and 1 have special meanings. Recommended to leave disabled to avoid unnecessarily reducing performance. 0x00: Disabled. READY will not necessarily deassert between frames. 0x01: Deassert READY after 256 consecutive frames. (Deasserting READY after just 1 frame can be accomplished by setting Synchronized Data Flow in the receiver, even if the transmitter is still using Pipelined Data Flow.) 0x02: Deassert READY after 2 consecutive frames. 0x03: Deassert READY after 3 consecutive frames. .. 0xFF: Deassert READY after 255 consecutive frames.

34.5.174 MIPIHSI_INTERRUPT_STATUS_0

HSI Interrupt Status Register

Offset: 0x3b4 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
6	X	TX_PIO_INTR: Asserts when TX_PIO_INT_STS bit in MISC_1_CONTROL Register is 1. 0 = NO_INTR 1 = INTR
5	X	RX_BREAK_FRAME_INTR: Asserts when RX_BREAK_FRAME_INT_STS bit in MISC_1_CONTROL Register is 1. 0 = NO_INTR 1 = INTR
4	X	RX_FRAME_TO_INTR: Asserts when RX_FRAME_TO_INT_STS bit in MISC_1_CONTROL Register is 1. 0 = NO_INTR 1 = INTR
3	X	RX_NEW_DATA_WR_COMP_INTR: Asserts when any bit of RX_NEW_DATA_WR_COMP_INTERRUPT_STATUS Register is 1. 0 = NO_INTR 1 = INTR
2	X	RX_ERROR_INTR: Asserts when any bit of RX_ERROR_INTERRUPT_STATUS Register is 1. 0 = NO_INTR 1 = INTR
1	X	RX_BUFFER_COMPLETION_INTR: Asserts when any bit of RX_DMA_BUFFER_COMPLETE_INTERRUPT_STATUS Register is 1. 0 = NO_INTR 1 = INTR

Bit	Reset	Description
0	X	TX_BUFFER_COMPLETION_INTR: Asserts when any bit of TX_DMA_BUFFER_COMPLETE_INTERRUPT_STATUS Register is 1. 0 = NO_INTR 1 = INTR

34.5.175 MIPIHSI_RX_FRAME_TIMEOUT_CNTR_0

HSI RX Frame Timeout Counter Register

Offset: 0x3b8 │ Read/Write: R/W │ Reset: 0x00000fff (0bxxxxxxxxxxxxxxxxxxxxx111111111111)

Bit	Reset	Description
11:0	0xfff	RX_FRAME_TIMEOUT: Receiver Frame Timeout Counter Number of HSI controller clock cycles after a frame starts that the last piece of data for that frame is received. If not, software is interrupted to deal with the apparently lost frame. The RX clock is nominally the same frequency as the TX clock, but may have a lower frequency. The default value of 0xFFF cycles is long enough considering the minimum data rate of rx. 0x00: Disabled 0x01: Frame started is considered incomplete after 1 controller clock cycle 0x02: Frame started is considered incomplete after 2 controller clock cycles. 0xFFF: Frame started is considered incomplete after 4095 controller clock cycles

34.5.176 MIPIHSI_MISC_2_CONTROL_0

HSI Misc 2 Control Register

Offset: 0x3bc │ Read/Write: R/W │ Reset: 0x0000012c (0bxxxxxxxxxxxxxxxxxxxxxxxx100101100)

Bit	Reset	Description
8:0	0x12c	COALESCE_TIMEOUT_CNTR: Coalesce timeout counter This is used to attempt to coalesce burst even when there is no back pressure from AHB. Counter has 300 as the default value (8*(32+4+1) = 296) to attempt burst of 8. 300 internal controller clock cycles after the first word of a channel is received to attempt to coalesce with following words. Software can disable this feature by setting this counter value as 0.

[THIS PAGE INTENTIONALLY LEFT BLANK]

35.0 I2C CONTROLLER

The I^2C controller (I2C) implements an I^2C 3.0 specification-compliant I^2C master and slave controller. The I^2C controller supports multiple masters and slaves. It supports Standard mode (up to 100 Kbits/s), Fast mode (up to 400 Kbits/s), Fast mode plus (Fm+, up to 1 Mbits/s), and High-speed mode (up to 3.4 Mbits/s).

Tegra® 4 devices have five instances of this controller. All five instances have identical I2C master functionality.

The I^2C controller supports DMA over the APB bus, using the APB DMA controller, in addition to single byte transfers. The I^2C controller also supports packet mode transfers where the data to be transferred is encapsulated in a predefined packet format as payload and sent to the I^2C controller over APB bus via DMA. The header of the packet specifies the type of operation to be performed, the size and other parameters (described in detail in subsequent sections).

Features

- Supports Standard (up to 100 Kbits/s), Fast (up to 400 Kbits/s), Fm+ (up to 1 Mbits/s), and High-speed (up to 3.4 Mbits/s) modes of operation.
- Independent Master Controller and Slave Controller with separate register-based host interface
- Master supports clock stretching by the slave
- Supports one to eight-byte burst data transfers
- Supports 4 Kbyte transfers in packet mode. Can be extended beyond 4 Kbytes by breaking up transfers into multiple packets (in both DMA and non-DMA modes)
- Both master and slave support transactions with 7-bit or 10-bit addressing
- Master is capable of data transfers to/from two or more slaves consecutively with a repeated-start condition
- Fully programmable 7-bit or 10-bit address for the slave
- Supports bus clear operations to address SDA issues
- Supports general call addressing
- Supports recognition and transfer of data to peripherals that do not send an acknowledge (ACK)
- Slave can also be configured not to acknowledge
- Master supports packet based DMA

Clocking

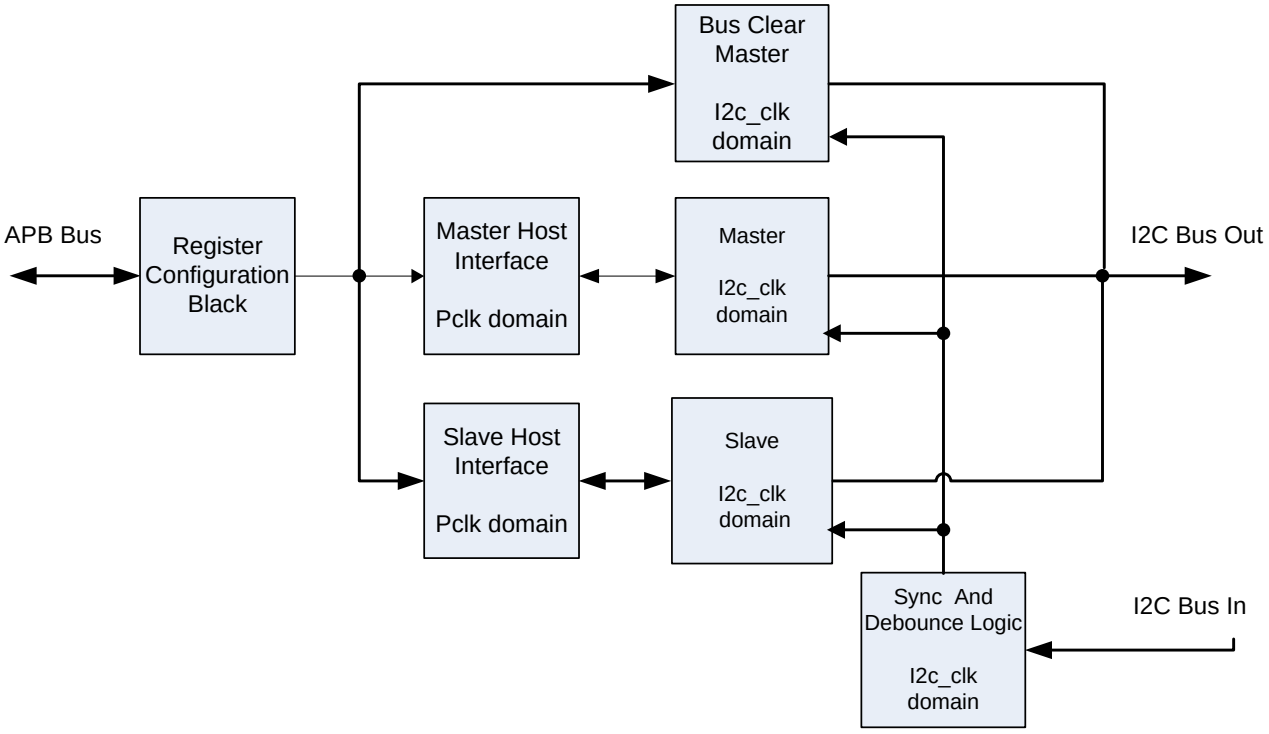
The I^2C controller has three clock domains:

- The Host interface runs on the APB clock (pclk). The APB clock is derived from the system clock and can be 1.0, ½, 1/3 or ¼ times the system clock (sclk).
- The I^2C interface controller runs on a fixed frequency clock running at 72 MHz. The 72 MHz clock is derived from PLLP. Even though you can mux between PLLP, PLLC, PLLM, and OSC clocks, PLLP is always selected and used in normal operation.
- The time-out logic runs on a 32 kHz clock.

35.1 Functionality

The I^2C controller can work as both a Master and a Slave. The Master can address the internal slave (for basic testing) or an external 7-bit or 10-bit addressed Slave device. The Master can be programmed for either a single slave transaction or a two-slave transaction. The two-slave transaction is generally useful in a random read from an external device. When reading from an external device, the random read-address needs to be set with an initial dummy-write to the device, followed by a repeated-start and a read transaction to the same device. The internal Slave address can be programmed to be either a 7-bit or a 10-bit address. Figure 140 below shows the I^2C controller in both the master and slave modes.

Figure 140: I²C Controller Block Diagram



35.1.1 Bus Clear Master Operation

Bus clear logic helps to resolve I2C bus hang issues. If the data line (SDA) is permanently stuck low because a slave device is pulling it low continuously for some unknown reason, the I2C master loses the arbitration and releases the I2C bus. In this case, software can use the Bus Clear master to get the SDA line released by sending clock pulses on the SCL line. Per the I2C specification, the device that held the bus Low should release it sometime within 9 clock pulses. But if the device needs more clock pulses than the default 9 pulses, software has an option to do so by programming a configurable register bit field (I2C_I2C_BUS_CLEAR_CONFIG_0[BC_SCLK_THRESHOLD]) to the required number of clock pulses.

Bus Clear Procedure

The Bus Clear operation starts with an ARB_LOST notification from the I^2C controller. The ARB_LOST notification could be due to other masters, if there are any master devices connected in the system and the I2C bus is already occupied by them. It could also be due to other configuration/setup issues such as pinmuxing, pin tri-state clear.

If it is determined that ARB_LOST notification is due to a slave device continuously pulling the SDA line low, the bus clear operation can be initiated to recover the bus with the following steps:

1. Reset the I^2C controller using module reset after the ARB_LOST notification from the controller.

2. Program the I2C_I2C_BUS_CLEAR_CONFIG_0[BC_SCLK_THRESHOLD] bit field to the required number of clock pulses.
3. Select the STOP or NO_STOP condition using the I2C_I2C_BUS_CLEAR_CONFIG_0[BC_STOP_COND] bit field.
4. Program I2C_I2C_BUS_CLEAR_CONFIG_0[BC_TERMINATE] as needed
5. Set the I2C_I2C_BUS_CLEAR_CONFIG_0[BC_ENABLE] bit field to start the bus clear operation. Hardware auto clears this bit after the bus clear operation is done.
6. Once enabled, the bus clear logic starts sending clock pulses on the SCL line.
7. Based on BC_TERMINATE settings, the Bus Clear logic:
   - Sends clock pulses until SDA is released or the threshold count is reached, whichever is earlier or
   - Sends the threshold number of clock pulses irrespective of SDA line release status (released before threshold is reached or not-released with threshold count reached)
8. If SDA is released within the programmed number of clock pulses:
   - The Bus Clear logic terminates the transaction with STOP/NO-STOP condition based on the BC_STOP_COND settings.
   - Then it sets the bus_clear operation status in the I2C_I2C_BUS_CLEAR_STATUS_0[BC_STATUS] register bit field.
   - Finally, it sets the BUS_CLEAR DONE bit field in I2C_INTERRUPT_STATUS_REGISTER_0 and also interrupts the system if the corresponding interrupt enable bit is set.

35.1.2 Low-Power Operation

I^2C is a low-speed serial interface with only two bus lines (SCL, SDA) required. So from the I^2C specification side, there is no low-power operation specifically defined for the interface. But to save the power, use the lowest possible speed, if performance is not needed. The following bus speeds are supported by I^2C:

- Standard mode (Sm), with a bit rate up to 100 kbit/s
- Fast mode (Fm), with a bit rate up to 400 kbit/s
- Fast mode Plus (Fm+), with a bit rate up to 1 Mbit/s
- High-speed mode (Hs mode), with a bit rate up to 3.4 Mbit/s

35.1.2.1 Low Power Programming

Program the CLK_SOURCE_I2C register (for example: I2C1instance clock source address 0x60006124) and the I2C_CLK_DIVISOR registers such that the required data rate can be achieved at the minimum possible module clock so that power can be saved.

Below are the data relations for various modes of operation. If the signal rise time is good in a system, use I2C_CLK_DIVISOR= 8 or above for Standard/Fast/Fm+ bus operations.

- Standard/Fast/Fm+:

  SCL = CLK_SOURCE.I2C / (N* (I2C_CLK_DIVISOR_STD_FAST_MODE+1)*I2C FREQUENCY DIVISOR)
- HS mode:

  SCL = CLK_SOURCE.I2C / (N * (I2C_CLK_DIVISOR_HSMODE+1)*I2C FREQUENCY DIVISOR)

The default N value is 8 for Standard/Fast/Fm+ modes and 12 for HS-mode. If there is a large rise time on the interface signals because of a weak pull-up or large capacitance, then N will increase. Also, N depends on I2C_CLK_DIVISOR_STD_FAST_MODE and I2C_CLK_DIVISOR_HSMODE values. It would be 9 and 13, respectively, for Standard and HS modes for I2C_CLK_DIVISOR_x values up to 2.

35.1.2.2 IDLE Power Programming

When the module is not used or the interface is idle, disable the module clock by clearing the CLK_ENB_I2C bits in the CAR module.

For example, the I2C1 clock enable bit is CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0[CLK_ENB_I2C1]. Software can clear this bit when it is seen that no transfers are needed from the I2C1 instance.

35.1.3 CL-DVFS-I2C and I2C5 sharing PWR-I2C Bus

Because the I2C5 controller and CL-DVFS-I2C can try to access the PWR-I2C bus simultaneously in an application, there is the potential for a bus conflict. To avoid this, the CL-DVFS module grants the interface bus to either the CL-DVFS-I2C or I2C5 controller based on the requests in round-robin fashion. If software intends to disable the CL-DVFS clock for some reason (for example, power savings), the CL-DVFS must be put in the RESET state by programming its module reset to 1. The I2C5 controller always gets the bus grant this way.

For the I2C5 controller to work properly, one of the following must occur:

1. The CL-DVFS is in the RESET state.
2. The CL-DVFS is out of reset and its clocks are running

The I2C5 controller will not work if the CL-DVFS is out of reset and the clocks to the CL-DVFS module are disabled. The interface bus grant arbiter in the CL-DVFS cannot work because its clock is not toggling.

35.1.4 TSEC-I2C and I2C4 Sharing DDC-I2C Bus

There is no bus arbiter in this case. Only one I2C (TSEC-I2C or I2C4) controller can access the DDC interface at a time. The controllers should be operated in serial fashion to access the DDC-I2C interface.

35.2 Software Interfaces

A typical peripheral controller might require the following steps to communicate or control an external peripheral:

1. Set up the registers
2. Program the ‘GO’ bit to start the interaction with the external peripheral
3. The peripheral controller collects the response from the peripheral and interrupts software
4. Software reads the response

35.2.1 Packet Flow

The flow is a transport path that can be uniquely identified in the system. It is a virtual path or channel that carries a particular type of packets from a source to a destination. A flow carries logical information and is not dependent on the physical implementation.

Information exchange between the hardware peripheral controller and software as described above can be classified into:

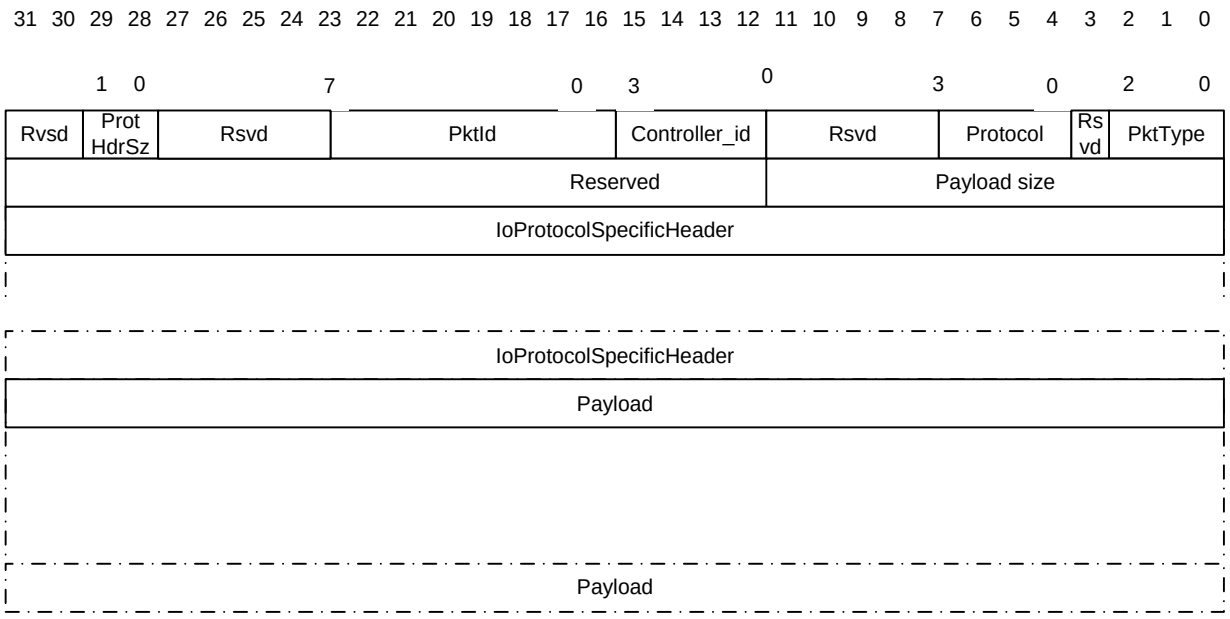
- ‘Request’ flow which can represent the register writes
- ‘Response’ flow for peripheral controller responses sent back to software, and
- ‘Interrupt’ flow for out-of-band handshakes between hardware and software

Packets within a flow can be exchanged in a manner that requires full intervention from the CPU (PIO mode) or least intervention from the CPU (DMA mode). The I/O packet described below is a structure that can be used to represent various flows in the system.

35.2.2 I/O Packet Format

An I/O packet consists of a header and payload. The header consists of two parts: two words of generic header and 1-4 words of protocol-specific header.

Figure 141: I/O Packet



35.2.2.1 I/O Packet Word 0

Word 0 of the I/O packet contains the Generic Header Word 0. The contents of generic header word 0 are described in the table below.

Table 107: Generic I/O Header Word 0

Bits	Field Value	Description
31:30	Reserved	Reserved
29:28	ProtHdrSz	Size of the protocol-specific header in words 0 = 1 words 1 = 2 words 2= 3 words 3 = 4 words For I2C, ProtHdrSz = 0 for request packets and 1 for response packets
27:24	Reserved	Reserved
23:16	PktId	PktId is an 8-bit field that identifies the packet. A peripheral controller might choose to log the packet ID for a packet that causes errors
15:12	Controller ID	There might be multiple controllers supporting any format. For example, there might be 4 I2C controllers. The Controller ID field specifies the controller for which this packet is destined.
11:8	Reserved	Reserved
7:4	Protocol	The protocol is indicated in this field 0 = reserved 1= I2C 2-15 = reserved
3	Reserved	Reserved

Bits	Field Value	Description
2:0	PktType	Packet type information 0 = request 1 = response 2 = interrupt 3 = stop 4-7 = reserved

35.2.2.2 I/O Packet Word 1

Word 1 of the I/O packet contains Generic Header Word 1. The contents of Generic Header Word 1 are described in the table below.

Table 108: Generic I/O Header Word 1

Bits	Field Value	Description
31:12	Reserved	Reserved
11:0	PayloadSize	Payload size in bytes

Note about Payload Size:

For request packets that transmit data, the total packet size is payload_size + 8 bytes of generic header + protocol-specific header. However, for request packets with a read command, the payload size indicates the amount of data to be received. Hence the packet size = 8 bytes of generic header + protocol-specific header, because there is no payload in this packet, just the receive command.

35.2.3 Transferring Packets

Packets can be transferred using PIO or DMA modes. A peripheral controller can choose to implement one or both modes. The header is a minimum of three words with the first two words reserved for packet information and the third word reserved for protocol-specific requirements. A minimum of four words is needed to transfer one byte to the external peripheral.

35.2.4 Protocol-Specific Header

Depending on the implementation of the peripheral controller and the interface protocol, the header words can be defined. No restriction is placed on the organization of the fields or the number of protocol-specific words (a minimum of one word and a maximum of four words) that a particular implementation chooses.

I2C has one word with the protocol-specific header. The contents of the I2C protocol-specific header are described in the next section.

35.2.5 I2C Master Packet Protocol Specific Headers

The I2C master supports two kinds of packets.

- Transmit packets that flow from Host to Controller.
- Receive data packets that flow from Controller to Host.

Table 109: I2C Master Transmit Packet Header

Bit	Name	Description
31:26	Reserved	Reserved
25	RESP_PKT_FREQ	Response packet frequency. This field is valid only if the I2C master is transmitting (READ/WRITE field below). If the I2C master is receiving, response packets are sent for every request packet received. 0: Send response packet at the end of last packet. 1: Send response packet for each packet transmitted.
24	RESP_PKT_ENABLE	Enable response packet. It qualifies RESP_PKT_FREQ field. 0: Do not send response packets (status has to be obtained via interrupt status bits). 1. Send response packet.
23	Reserved	Reserved
22	HS_MODE	Enable High speed mode (3.4 MHz operation)
21	CONTINUE_ON_NACK	Enable mode to handle devices that do not generate ACK upon the reception of a byte.
20	SEND_START_BYTE	1: Send a start byte at the beginning of the transaction
19	READ/WRITE	1: READ
18	Address mode	1: 10-bit mode 0: 7-bit mode
17	IE	Generate interrupt upon packet completion
16	REPEAT_START/STOP	1: Put a repeat start condition on the bus(to continue transaction) 0: Put a stop condition on the bus
15	ContinueXfer	This bit overrides the REPEAT_START/STOP command above. 0: Introduce repeat start or stop condition based on bit 16. 1: Continue with current transfer without stop or repeat start condition.
14:12	HS_MASTER_ADDR	High Speed mode Master code
11:10	Reserved	Reserved
9:0	SLAVE_ADDR	Slave address. Bit 0 is ignored for 7-bit addressing, but should always match bit 19 (READ/WRITE).

Response packets contain the status of the I^2C controller. To accommodate that, the response packet header contains two words of the protocol-specific header.

Table 110: I2C Master Response Packet Header Word 0

Bit	Name	Description
31:27	Reserved	Reserved
26	RFIFO_OVF	Receive FIFO overflowed.
25	TFIFO_OVF	Transmit FIFO overflowed.
24	TRANSFER_COMPLETE	All the packets have been transferred successfully. The packet for which the “LAST PACKET” field is set has been transferred.
23:16	TRANSFER_PKT_ID	The current packet ID for which the transaction is happening on the bus
15:4	TRANSFER_BYTENUM	The number of bytes transferred in the current packet

Bit	Name	Description
3	NOACK_FOR_ADDR	1 = No ACK received for the address byte
2	NOACK_FOR_DATA	1 = No ACK received for the current data byte
1	ARB_LOST	1 = Arbitration lost
0	CONTROLLER_BUSY	1 = Controller is busy

35.2.6 I2C Slave Protocol Specific Packet Headers

The I2C slave supports two packet types:

- Transmit packets flow from the Host to I^2C controller. These contain details of the transfer that need to be carried out, such as transfer direction (read/write) and mode of operation.
- Receive packets flow from the I^2C controller to Host. They contain the data (in case of the I2C receiving data) as well as the status that indicates the health of the I2C slave.

Note: For I2C slaves, the transmit packet can be of control type (generic header word 0, bits [2:0]). This can be used by the host to tell the I2C slave to acknowledge the last byte of the transfer.

Table 111: I2C Slave Transmit Packet Header

Bit	Name	Description
31:26	Reserved	Reserved
26	ACK_LAST_BYTE	This bit is set to indicate to the I2C slave that it can release the bus by acknowledging the last byte. 0: Withhold ACK for the last byte of transfer. 1: Acknowledge the last byte immediately.
25	RESP_PKT_FREQ	Response packet frequency. This field is valid only if the I2C Slave is transmitting. If the I2C slave is receiving, then it will send a response packet for each request packet received. 0: Send response packet at the end of last packet 1: Send response packet for each packet transmitted.
24	RESP_PKT_ENABLE	Enable response packet. 0: Do not send response packets (status has to be obtained via interrupt status bits). 1: Send response.
23	Reserved	Reserved
22	HS_MODE	Enable High speed mode (3.4 MHz operation)
21:20	Reserved	Reserved
19	READ/WRITE	1: READ
18	Address mode	1: 10-bit mode 0: 7-bit mode
17	IE	Generate interrupt upon packet completion
16:11	Reserved	Reserved.
10	SLAVE_ADDR_OVRIDE	Use the slave address field in the packet header (below) instead of the programmed value.
9:0	SLAVE_ADDR	Slave address

The I2C Slave Response Packet protocol-specific header contains two words.

Note: Slave response packet header word 0 contains what might seem redundant information; however, this information is useful if the external master operates in a mode different from what the host expects. For example, the external master while capable of High Speed mode can operate in Fast mode, which is helpful for the host to know.

The same is true for the slave address, because two slave addresses are supported.

Table 112: I2C Slave Receive Packet Header Word 0

Bit	Name	Description
31:23	Reserved	Reserved
22	HS_MODE	Enable High speed mode (3.4 MHz operation)
21:19	Reserved	Reserved
18	Address mode	1 = 10-bit mode 0 = 7-bit mode
17:10	Reserved	Reserved
9:0	SLAVE_ADDR	Slave address

Table 113: I2C Slave Receive Packet Header Word 1

Bit	Name	Description
31:30	Reserved	Reserved
29	ACK_WITHHELD	Indicates that the ACK is withheld for the last byte and the slave is waiting for the host to explicitly command the slave to acknowledge the last byte. 0: Bus is released. 1: ACK is withheld.
28:27	XFER_ERR_REASON	This bit describes the nature of packet transfer error. It is meaningful only if PKT_XFER_ERR is set. 0: Master terminated transaction before it was completed. 1: Master did not terminate transaction when all bytes were transferred. 2: Reserved 3: Reserved
26	PKT_XFER_ERR	0: Request was successful 1: Error has occurred during packet transfer.
25	TFIFO_OVF	Transmit FIFO overflowed
24	TFIFO_OVF	Transmit FIFO overflowed
23:16	TRANSFER_PKT_ID	The current packet ID for which the transaction is happening on the bus
15	Reserved	Reserved
14:8	HW_MS_ADDR	Hardware master address (received via hardware GCA)
7	Reserved	Reserved
6	HS_MST_IRQ	Slave received hardware master address via GCA
5	REPROG_SL	Slave received ReprogramSlaveAddress command via GCA (general call addressing)
4	RST_SL	Slave received ResetSlave command via GCA (general call addressing)
3:2	Reserved	Reserved
1	R/W	Slave Transaction Status 1: Read transaction happened. 0: Write to slave happened.
0	ZA	Zero Address Status: Reports if slave has responded to the general call address (zero address). Valid only during writes. 1: Yes, slave responded 0: No, slave did not respond

35.3 Programming Guidelines

35.3.1 I2C Frequency Divisor Register

The FREQUENCY DIVISOR register (CLK_SOURCE_I2C register, whose address is: 0x60006124 for I2C1) must be programmed as a function of the CLK_SOURCE selected for I2C as follows:

```
Standard/Fast-mode/Fm+:
SCL = CLK SOURCE.I2C / (N* (I2C CLK DIVISOR STD FAST MODE+1)*I2C FREQUENCY DIVISOR)
High Speed mode:
SCL = CLK SOURCE.I2C / (N * (I2C CLK DIVISOR HSMODE+1)*I2C FREQUENCY DIVISOR)
Note:  The default N value is 8 for Standard/Fast/Fm+ modes and 12 for High Speed mode.
If there is a large rise time on the interface signals because of a weak pull-up or large capacitance, N
will increase. Also, N depends on I2C CLK DIVISOR STD FAST MODE and I2C CLK DIVISOR HSMODE values. It
would be 9 and 13, respectively, for Standard and High Speed modes for I2C_CLK_DIVISOR_x values up to 2.
```

The I^2C bus specification defines the minimum low period for the I2C_CLK as 4.7µs in Standard mode and 1.3µs in Fast mode. Because of this, the maximum I2C_CLK frequency in the standard mode can be 100 KHz but in Fast mode, it is limited to 348 KHz, assuming I2C_CLK has rise and fall delays of 300 ns per the I^2C specification.

The clock enable (bit 12 of the CLK_OUT_ENB.L register) must also be given to the I^2C controller, before any of the registers are written.

35.3.2 I2C Slave ADDR Registers

There are two slave ADDR registers that are used to configure the address of the internal slave-- I2C_SL_ADDR1 and I2C_SL_ADDR2. Bit [0] of I2C_SL_ADDR2 determines the address size of the internal slave. When this bit is programmed to be zero, the 7-bit slave address is taken from I2C_SL_ADDR1 [6:0]. When this bit is programmed to be one, the two most significant bits of the 10-bit slave address are taken from I2C_SL_ADDR2[2:1], and the least significant bits are taken from I2C_SL_ADDR1[7:0].

35.3.3 I2C Command ADDR Registers

These registers, I2C_CMD_ADDR0 and I2C_CMD_ADDR1, contain the address of the slave with which a transaction is intended. The I^2C Master Controller can be programmed to transact with both 7-bit and 10-bit addressed slaves. Bit [0] of the I2C_CNFG register determines the size of the slave address to be transacted. If a 7-bit or a 10-bit slave address is chosen, the respective address is taken from I2C_CMD_ADDR0 [9:0], and the Read/Write command is taken from bits 6 and 7 of the I2C_CNFG Register. In a 2 Slave configuration, the slave 1 address (7-bit or 10-bit) is taken from I2C_CMD_ADDR0 [9:0] and Slave 2 address (7-bit or 10-bit) from I2C_CMD_ADDR1 [9:0].

35.3.4 I2C Data Registers

There are two data registers, I2C_CMD_DATA1and I2C_CMD_DATA2, each with 32-bit length, which are to be loaded with the data to be transmitted or received as an I^2C Master. When used as an I2C slave, the controller uses a separate byte-wide register to store the data to be transmitted (I2C_SL_SEND) or data received (I2C_SL_RCVD).

35.3.5 I2C Configuration Register

This register is to be configured with the following data:

- Number of bytes to be transmitted or received
- Select for 7-bit or 10-bit slave address
- Program to send a Start Byte
- Single or two slave operations
- Support of NOACK from an external peripheral

The I2C_CNGF [9] bit is used to issue a “Begin-Transaction” command to the I^{2}C Master Controller. This bit is reset by hardware, and other bits of the register are masked for writes, when this bit is programmed to be one. Hence, the firmware should first configure all the other registers and bits [8:0] of the I2C_CNFG register before the I2C_CNFG [9] bit is programmed to one.

35.3.6 I2C Controller Status Register

This register (I2C_STATUS) gives the status of the I^{2}C Master operation. It includes the busy status of the I^{2}C controller and the completion status of the command executed.

35.3.7 I2C Controller Slave Status Register

This register (I2C_SL_STATUS) contains the status bits as I^{2}C Slave.

The controller contains status bits as follows:

- Receive Interrupt
- New Transaction Received
- Direction of transfer
- Bit 4 contains the END_TRANS bit. This bit is set when an I^{2}C transfer is complete (as indicated by either a stop or repeat start condition).
- Bit 5 contains the RST_SL bit. The byte received following the receipt of the general call address is 0x06. Therefore the programmable part of the slave address needs to be reset and reprogrammed.
- Bit 6 contains the REPROG_SL bit. The byte received following the receipt of the general call address is 0x04. Therefore reprogram the programmable part of slave address.
- Bit 7 contains the HW_MSTR_INT bit. When set, this bit indicates that master address has been transmitted after the general call address.
- Bits [14:8] contain the hardware master address. The contents of this field are meaningful only when bit 7 (HW_MSTR_INT) is set.

The I^{2}C controller generates an interrupt at the completion of each data byte received. In response to this interrupt, firmware must read the I2C_SL_STATUS register to confirm the cause of the interrupt and the direction of data transfer. If the interrupt is due to a data byte received (R/W = 0), firmware must read the data register, I2C_SL_RCVD, in order to clear the SL.IRQ bit in the status register. The I^{2}C controller will delay the release of the ACK handshake on the I^{2}C bus until the SL.IRQ bit has been cleared by this firmware read operation.

Either I2C_SL_STATUS needs to be polled or the I2C interrupt must be enabled for transfers to occur in slave mode. The following steps outline slave operation in both read and write transactions:

If the controller is configured as a slave, and is receiving data,

- I2C_SL_STATUS.SL_IRQ bit being set indicates a data byte has been received.
- Read of I2C_SL_STATUS clears SL_IRQ and returns the transfer direction in the RNW field.
- The data byte is already present in the I2C_SL_RCVD register. This can be read which causes the bus to be released.
- SL_IRQ is set again upon reception of the next byte.

If the controller is configured as a slave, and is transmitting data,

- I2C_SL_STATUS.SL_IRQ bit being set indicates that an address has been received.
- Read of I2C_SL_STATUS clears SL_IRQ and returns the transfer direction in the RNW field.
- The data byte needs to be written to the I2C_SL_RCVD register. This write causes the bus to be released.
- SL_IRQ is set again upon transfer of the byte. Another byte can be written to continue the operation.

35.3.8 Interrupt Generation

The I^2C controller generates interrupts upon the completion of transmission or reception of the specified number of bytes. A Master interrupt is generated when the number of bytes in the transaction, as programmed in the LENGTH field of the I2C_CNFG register, is complete.

A Slave interrupt is generated at the transmission/reception of each byte of data by the internal slave. In addition, if the internal Slave is in transmitter mode (the Slave receives a Read command), an interrupt is generated for the address and read-command reception also, as shown in the next two figures below. Slave interrupts are to be cleared by the firmware as mentioned above.

Figure 142: Interrupt Generation: Master Transmits to Internal Slave

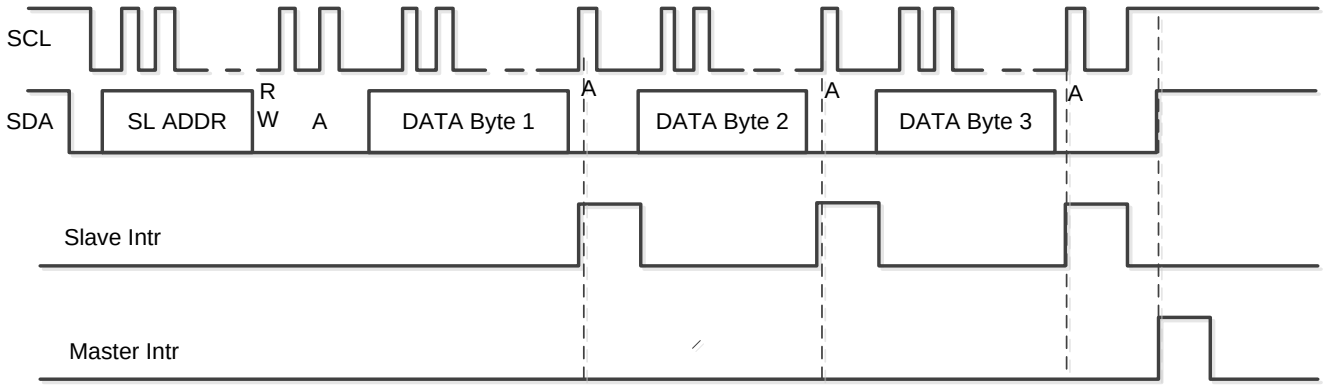
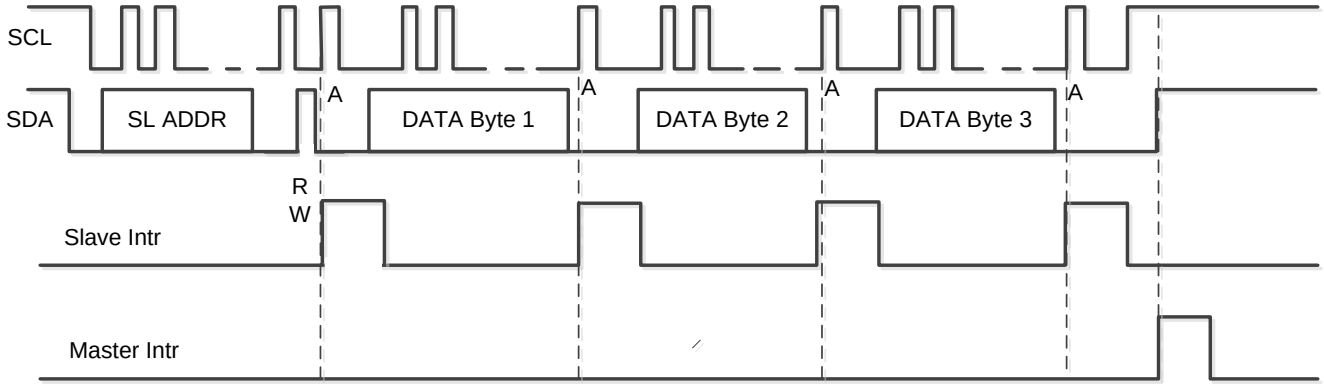


Figure 143: Interrupt Generation: Master Reads from Internal Slave



35.4 Programming Guidelines for Packet-Based Interface

35.4.1 Packet-Based Interface Registers

The following registers need to be programmed for a packet-based interface:

- I2C TX Packet FIFO
- I2C RX FIFO
- PACKET TRANSFER STATUS
- FIFO CONTROL
- FIFO STATUS
- Slave I2C TX FIFO
- Slave I2C RX FIFO
- Slave PACKET_TRANSFER_STATUS
- Slave FIFO CONTROL
- FIFO STATUS (common for both master and slave)
- INTERRUPT MASK REGISTER (common for both master and slave)
- INTERRUPT STATUS REGISTER (common for both master and slave)
- PACKET_MODE_EN, DEBOUNCE_CNT, and NEW_MASTER_FSM fields of I2C Controller Configuration Register

All other registers/fields have no meaning in packet mode and should be used only normal mode.

35.4.2 Programming Packet Header and Payload

Software should program the I2C_TX_PACKET_FIFO register. It writes the packet header followed by the payload. The header size can vary from 3 to 5 words. For I2C, the size is 3 words for request packets and 4 words for response packets. The first two words of the header contain a generic header. The third word of the packet (and fourth as well for response packets) contains I2C transaction-specific information. The payload contains the actual data to be written to the slave. In case of read operations, payload is nil, and hence, the packet contains only a header.

35.4.3 Reading from the RX FIFO

The data received on the bus is pushed into the RX FIFO. In PIO mode, a request interrupt is generated when the attention level is reached, and software reads the RX_FIFO register to get the data. In DMA mode, a DMA trigger is asserted after reaching the attention level.

35.4.4 Error Handling

The following steps describe what needs to happen when an error occurs due to a NACK, loss of arbitration, TX FIFO overflow, or RX FIFO overflow before proceeding with any further transactions:

- When an error occurred due to a NACK, a stop condition is put on the bus and interrupt is generated
- The status register is updated with the current error packet ID at which the error occurred
- Software should reset the controller
- In case of a DMA transfer, DMA needs to be restarted

In packet mode, special care should be taken to handle error recovery. If an error occurs during the transfer of a packet, then the controller should be reset, DMA needs to be restarted (if using DMA transfers), and the entire packet needs to be resent since there is no accurate way of knowing how many bytes made it to the I2C slave.

The repeat start bit in packet mode can complicate the situation further. When back-to-back packets are sent to a slave with the repeat start bit set, and an error occurs, it is not always known during which packet transfer the error occurred. Therefore, the entire stream of packets that were sent back to back using the repeat start bit need to be sent again.

Receives are simpler, since if an error occurs, and bytes for a packet have already been received, that packet can be deemed complete. Only incomplete transfers need to be retried.

35.4.5 Programming repeat_start/stop

This field indicates whether or not to put a stop condition after the current transaction. By default, this bit is zero and a stop condition is put on the bus. If this bit is set, a repeat start is put on the bus before proceeding with the next packet. The Repeat_start/stop bit can be used for combining read operations and write operations within a single transaction with a repeat start, or to do transfers beyond the 4 Kbyte limit. This bit is present in the protocol-specific header.

The I^2C controller supports transfer sizes beyond 4 Kbytes without a repeat start condition, again by combining multiple packets. This is in addition to ability to use repeat start condition to do continuous transfers over 4 Kbyte limit.

35.4.6 FIFO Control

In DMA mode, TX_FIFO_TRIG indicates the number of words that need to be empty in the TX FIFO for the DMA trigger to be asserted. RX_FIFO_TRIG indicates the number of words that need to be full in the RX FIFO for the DMA trigger to be asserted.

In PIO mode, TFIFO_DATA_REQ in the Interrupt Status Register will be set if the TX FIFO empty count is more than the value programmed in TX_FIFO_TRIG. Similarly RFIFO_DATA_REQ in the Interrupt Status Register will be set if the RX FIFO full count is more than the value programmed in RX_FIFO_TRIG. The CPU will be interrupted if the status bits are set and their corresponding INT_EN bit is set in the Interrupt Mask Register. The depth of the TX and RX FIFOs is 8.

35.4.7 FIFO Status

This register indicates the number entries that are empty in the TX FIFO (TX_FIFO_EMPTY_CNT) and the number of entries that are full in the RX FIFO(RX_FIFO_FULL_CNT).

35.4.8 Example Transfer

Here is a programming example for a 1 byte read followed by a 5-byte write

1. Program the PACKET_MODE_EN field to 1
2. Write the generic packet header
3. Write the payload size = 0x0 (1 byte)
4. Write the I2C specific header read/write = 1 and repeat_start/stop =1
5. Write the generic packet header
6. Write payload size = 0x4 (5 bytes)
7. Write the I2C specific header read/write = 0 and repeat_start/stop
8. Write the 12 bytes (word aligned)

35.4.9 Removal of Register Read Side Effects

The I^2C controller implements packet mode transfers for I2C slaves. In packet mode, the slave holds off sending an ACK on the very last byte of transfer until the host explicitly indicates to the I^2C controller to release the bus by acknowledging the last byte. The Host is responsible for providing this indication to the I2C slave in a timely fashion before the SMBUS time-outs happens.

After checking the packet contents, the Host needs to send a control packet with:

- ACK_LAST_BYTE in the protocol-specific header set to 1'b1
- PAYLOAD_SIZE in the generic packet header set to zero
- READ/WRITE in the protocol-specific header set to 1'b1 (READ).

The slave will release the bus as soon as it receives this control packet.

35.4.10 DMA Capability for I2C Slave

Packet mode capability of the slave combined with DMA transfer capabilities eliminates the need for byte-by-byte interactions between the driver and hardware.

35.4.11 Programming Guidelines for the Slave

The data is transmitted/received through the I2C_I2C_SL_RCVD_0 register.

To work in a mode where one byte can be transferred at a time, set the ENABLE_SL bit in the I2C_I2C_SL_CNFG_0 register.

Data can be transmitted and received through the I2C_I2C_SLV_TX_PACKET_FIFO_0 and I2C_I2C_SLV_RX_FIFO_0 registers, respectively when the FIFO_XFER_EN bit in the I2C_I2C_SL_CNFG_0 register is set.

In FIFO transfer mode, a slave by default works in one byte mode. That is, initially when a single byte is received, the ACK is withheld and an interrupt is generated. Software should indicate whether to acknowledge this byte or not by writing into ACK_LAST_BYTE field of the I2C_I2C_SL_CNFG_0 register. Note that the ACK_LAST_BYTE_VALID field also needs to be set to indicate to hardware that ACK_LAST_BYTE is being updated. When software receives an interrupt due to SLV_RX_BUFFER_FILLED, it can check ACK_WITHHELD of the I2C_I2C_SLV_PACKET_STATUS_0 register to see if the slave is withholding the ACK for the last byte.

Software can update the buffer size from default to any number by programming BUFFER_SIZE or by writing a packet where it updates the payload size.

The packet structure consists of header followed by data in case of a Read operation, whereas it consists of only header in case of a Write operation. Packets should be programmed into the I2C_I2C_SLV_TX_PACKET_FIFO_0 register .The data received will be stored in I2C_I2C_SLV_RX_FIFO_0 register. When software reads this register, it causes a pop.

The packet header consists of 3 words: a generic header, payload size, and a protocol-specific header.

A packet Write operation works as follows.

After enabling the packet, to work in packet mode for a Write operation (Master writing to slave):

- Set I2C_I2C_SL_CNFG_0.ENABLE_SL
- Set I2C_I2C_SL_CNFG_0.FIFO_XFER_EN
- Set I2C_I2C_SL_CNFG_0.PACKET_MODE_EN
- Wait for an SLV_RX_BUFFER_FILLED interrupt. By default, buffer_size is one. An interrupt will be generated after the first byte and the ACK is withheld.
- Clear the interrupt and indicate whether to acknowledge the last byte received by writing into the ACK_LAST_BYTE_VALID and ACK _LAST_BYTE bits of the I2C_I2C_SL_CNFG_0 register.
- Write the Generic Packet Header
- Write the Payload size
- Write the protocol-specific header.

The buffer_size will be updated with the payload size set in packet header. An interrupt will be generated again after receiving the number of bytes equal to buffer_size. The Slave will withhold the ACK for the last byte if the ACK_LAST_BYTE field in the protocol-specific header is not set or else it will acknowledge the last byte immediately.

Software should read the bytes received from SLV_RX_FIFO register. This will generate a trigger if the FIFO is above the threshold. See the FIFO_CONTROL register on how to program threshold levels. The last two steps are repeated every time there is a transfer equal to buffer_size.

To work in packet mode for Read operations (Master reading from slave):

- Set I2C_I2C_SL_CNFG_0.ENABLE_SL
- Set I2C_I2C_SL_CNFG_0.FIFO_XFER_EN
- Set I2C_I2C_SL_CNFG_0.PACKET_MODE_EN
- When a read transaction is received, an interrupt will be generated and TX_BUFFER_REQ will be set.
- Clear the interrupt and write the data to be transmitted into the I2C_I2C_SLV_TX_PACKET_FIFO_0.
- If a new transaction is received, an interrupt will again be generated and TX_BUFFER_REQ will be set.

The data can be sent or received without working in packet mode by only updating the following register field information. For example, a Write operation works as follows:

- Set I2C_I2C_SL_CNFG_0.ENABLE_SL
- Set I2C_I2C_SL_CNFG_0.FIFO_XFER_EN
- Set I2C_I2C_SL_CNFG_0.ACK_WITHHOLD_EN if hardware needs to withhold the ACK for the last byte.
- After receiving SLV_RX_BUFFER_FILLED interrupt, clear the interrupt and update the ACK_LAST_BYTE field. Also read the bytes from RX_FIFO.

35.5 I2C Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

35.5.1 I2C_I2C_CNFG_0

IC Controller Configuration Register (Master)

The I2C_CNFG register is used to configure:

- the number of bytes to be transmitted or received
- the slave device type (either a 7-bit device or a 10-bit device)
- enable mode to send Start-byte or not
- to select either a single slave transaction or two slave transaction
- enable mode to handle devices that do not generate an ACK.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000800 (0bxxxxxxxxxxxxxxxx0000100000000000)

Bit	Reset	Description
15	0x0	MSTR_CLR_BUS_ON_TIMEOUT: When this bit is set, the I2C master will force the clock low for an extended period of time (>TIMEOUT) to force all SMB slaves to release the bus. 0 = Don’t clear the bus on TLow: SEXT/TIMEOUT time-out 1 = Clear the bus on TLow: SEXT/TIMEOUT time-out

Bit	Reset	Description
14:12	0x0	DEBOUNCE_CNT: Debounce period for SDA and SCL lines. 0 = No debounce 1 = 2T 2 = 4T 3 = 6T, etc. where T is the period of the fix PLL clock source coming to I2C. Maximum debounce period is 14T. A debounce period of > 50 ns is desirable.
11	0x1	NEW_MASTER_FSM: Maintained for backward compatibility. 0 = DISABLE 1 = ENABLE
10	0x0	PACKET_MODE_EN: Write 1 to initiate transfer in packet mode. 0 = NOP 1 = GO
9	0x0	SEND: Writing a 1 causes the master to initiate the transaction in normal mode. Values of other bits are not affected when this bit is 1. Cleared by hardware. Other bits of the register are masked for writes when this bit is programmed to one. Thus, firmware should first configure all other registers and bits [8:0] of the I2C_CNFG register before the bit I2C_CNFG[9] is programmed to zero. 0 = NOP 1 = GO
8	0x0	NOACK: Enable mode to handle devices that do not generate an ACK. 1 = Do not look for an ACK at the end of the enable. 0 = DISABLE 1 = ENABLE
7	0x0	CMD2: Read/Write Command for Slave 2: 1 - Read Transaction; 0 - Write Transaction. For a 7-bit slave address, this bit must match with the LSB of address byte for slave 2. Valid only when bit 4 of this register is set. 0 = DISABLE 1 = ENABLE
6	0x0	CMD1: Read/Write Command for Slave 1: 1 - Read Transaction; 0 - Write Transaction. Command for Slave 1: For a 7-bit slave address, this bit must match with the LSB of address byte for slave 1. 0 = DISABLE 1 = ENABLE
5	0x0	START: 1 = Yes, a Start byte needs to be sent. 0 = DISABLE 1 = ENABLE
4	0x0	SLV2: 1 = Enables a two-slave transaction. 0 = No command for Slave 2 present. 0 = DISABLE 1 = ENABLE
3:1	0x0	LENGTH: The number of bytes to be transmitted per transaction. 000 = 1 byte ... 111 = 8 bytes. In a two slave transaction, the number of bytes should be programmed to be less than 011.
0	0x0	A_MOD: Address mode defines whether a 7-bit or a 10-bit slave address is programmed. 1 = 10-bit device address. 0 = 7-bit device address. 0 = SEVEN_BIT_DEVICE_ADDRESS 1 = TEN_BIT_DEVICE_ADDRESS

35.5.2 I2C_I2C_CMD_ADDR0_0

I2C Slave-1 Address

I2C_CMD_ADDR0 is programmed with the 7-bit or 10-bit address of slave 1 to which the transaction is intended.

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	ADDR0: For 7-bit mode, the address is written in I2C_CMD_ADDR0[7:1], and I2C_CMD_ADDR0[0] indicates the read/write transaction. The I2C_CMD_ADDR0[0] bit must match the I2C_CNFG[6] bit. For 10-bit mode, the address is written in I2C_CMD_ADDR0[9:0], and I2C_CNFG[6] indicates the read/write transaction.

35.5.3 I2C_I2C_CMD_ADDR1_0

I2C Slave-2 Address

I2C_CMD_ADDR1 is programmed with the 7-bit or 10-bit address of slave 2 to which the transaction is intended.

Offset: 0x8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0000000000)

Bit	Reset	Description
9:0	0x0	ADDR1: For 7-bit mode, the address is written in I2C_CMD_ADDR0[7:1], and I2C_CMD_ADDR0[0] indicates the read/write transaction. The I2C_CMD_ADDR0[0] bit must match the I2C_CNFG[7] bit. For 10-bit mode, the address is written in I2C_CMD_ADDR0[9:0], and I2C_CNFG[7] indicates the read/write transaction.

35.5.4 I2C_I2C_CMD_DATA1_0

I2C Controller Data 1: Transmit/Receive

The four least significant bytes of data to be transmitted are loaded into this register when the I2C Master is in Write mode.

The four least significant bytes of data are read through this register when the I2C Master is in Read mode.

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

Bit	Reset	Description
31:24	0x0	DATA4: Fourth data byte to be sent/received
23:16	0x0	DATA3: Third data byte to be sent/received
15:8	0x0	DATA2: Second data byte to be sent/received
7:0	0x0	DATA1: First data byte to be sent/received

35.5.5 I2C_I2C_CMD_DATA2_0

I2C Controller Data 2: Transmit/Receive

The four most significant bytes of data to be transmitted are loaded into the register when the I2C Master is in Write mode.

The four most significant bytes of data are read through this register when the I2C Master is in Read mode.

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	DATA8: Eighth data byte to be sent/received
23:16	0x0	DATA7: Seventh data byte to be sent/received
15:8	0x0	DATA6: Sixth data byte to be sent/received
7:0	0x0	DATA5: Fifth data byte to be sent/received

35.5.6 I2C_I2C_STATUS_0

I2C Controller Status (Master)

I2C_STATUS gives the status of the I2C Master operation.

Offset: 0x1c | Read/Write: RO | Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	BUSY: 0 = NOT_BUSY 1 = BUSY
7:4	X	CMD2_STAT: Slave 1 status: Transaction for Slave 1 for byte x failed. x is 'h0 to 'ha. All other combinations are invalid. 0 = SL1_XFER_SUCCESSFUL 1 = SL1_NOACK_FOR_BYTE1 2 = SL1_NOACK_FOR_BYTE2 3 = SL1_NOACK_FOR_BYTE3 4 = SL1_NOACK_FOR_BYTE4 5 = SL1_NOACK_FOR_BYTE5 6 = SL1_NOACK_FOR_BYTE6 7 = SL1_NOACK_FOR_BYTE7 8 = SL1_NOACK_FOR_BYTE8 9 = SL1_NOACK_FOR_BYTE9 10 = SL1_NOACK_FOR_BYTE10
3:0	X	CMD1_STAT: Slave 2 status: Transaction for Slave 2 for byte x failed. x is 'h0 to 'ha. All other combinations are invalid. 0 = SL2_XFER_SUCCESSFUL 1 = SL2_NOACK_FOR_BYTE1 2 = SL2_NOACK_FOR_BYTE2 3 = SL2_NOACK_FOR_BYTE3 4 = SL2_NOACK_FOR_BYTE4 5 = SL2_NOACK_FOR_BYTE5 6 = SL2_NOACK_FOR_BYTE6 7 = SL2_NOACK_FOR_BYTE7 8 = SL2_NOACK_FOR_BYTE8 9 = SL2_NOACK_FOR_BYTE9 10 = SL2_NOACK_FOR_BYTE10

35.5.7 I2C_I2C_SL_CNFG_0

I2C Controller Configuration (Slave)

I2C_SL_CNFG register is used to configure, enable mode of slave ACK.

Enable mode of the slave response to the general call address.

The register should be programmed when the I^2C controller is configured as a slave.

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxx00000000000000000100)

Bit	Reset	Description
20	0x0	FIFO_XFER_EN: If this bit is disabled, data is always communicated via the I2C_SL_RCVD register. If enabled, data is communicated through FIFOs 0 = DISABLE 1 = ENABLE
19:8	0x0	BUFFER_SIZE: Payload size in bytes
7	0x0	ACK_LAST_BYTE_VALID: Acknowledge the last byte valid (write-only). This bit qualifies the ACK_LAST_BYTE field. 0 = DISABLE 1 = ENABLE
6	0x0	ACK_LAST_BYTE: Acknowledge the last byte. 0 = DISABLE 1 = ENABLE
5	0x0	ACK_WITHHOLD_EN: ACK Withhold Feature Enable 0 = DISABLE 1 = ENABLE
4	0x0	PKT_MODE_EN: Packet Mode Enable 0 = DISABLE 1 = ENABLE
3	0x0	ENABLE_SL: By writing zero to this field, the slave can be turned off 0 = DISABLE 1 = ENABLE
2	0x1	NEWSL: New Slave. 1 - Use new slave. 0 = DISABLE 1 = ENABLE
1	0x0	NACK: Disable Slave ACK. 1 – The slave will not acknowledge reception of address or data byte. 0 = DISABLE 1 = ENABLE
0	0x0	RESP: Slave response to the general call address (zero address). 0 = DISABLE 1 = ENABLE

35.5.8 I2C_I2C_SL_RCVD_0

I2C Controller Slave Receive/Transmit Data (Slave)

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SL_DATA: Slave Received data

35.5.9 I2C_I2C_SL_STATUS_0

I2C Controller Slave Status (Slave)

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000XX0X (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000x0)

Bit	R/W	Reset	Description
14:8	RO	X	HW_MSTR_ADR: Hardware master address received via general call addressing. This field is meaningful only if HW_MSTR_INT is set.
7	RW	0x0	HW_MSTR_INT: 1 = Interrupt has been generated by slave. Hardware Master Address is received after General Call Address. Received Hardware Master Address. 0 = No event.
6	RW	0x0	REPROG_SL: 1 = Interrupt has been generated by the slave after the General Call Address is 0x04. Reprogram slave address. 0 = No action.
5	RW	0x0	RST_SL: 1 = Interrupt has been generated by slave after the General Call Address is 0x06. Reset and reprogram slave address. 0 = No action.
4	RW	0x0	END_TRANS: 1 = Interrupt has been generated by slave. Transaction completed as indicated by stop/repeat start condition. 0 = No transaction occurred or transaction in progress.
3	RW	0x0	SL_IRQ: 1 = Interrupt has been generated by slave. 0 = No interrupt generated. 0 = UNSET 1 = SET
2	RW	0x0	RCVD: New Transaction Received status. 1 = Transaction occurred. 0 = No transaction occurred. 0 = NO_TRANSACTION_OCCURED 1 = TRANSACTION_OCCURED
1	RO	X	RNW: Slave Transaction status. 0 = WRITE 1 = READ
0	RW	0x0	ZA: Zero Address Status 1 = Yes, slave responded. 0 = No, slave did not respond. 0 = NO_SLAVE_RESPONSE 1 = SLAVE_RESPONSE

35.5.10 I2C_I2C_SL_ADDR1_0

I2C Controller Slave Address 1 Register (Slave)

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:8	0x0	SL_ADDR1: For a 10-bit slave address, this field is the least significant 8 bits.
7:0	0x0	SL_ADDR0: For a 10-bit slave address, this field is the least significant 8 bits.

35.5.11 I2C_I2C_SL_ADDR2_0

I2C Controller Slave Address 2 Register (Slave)

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0xxxxx000xxxxx000)

Bit	Reset	Description
16	0x0	SELECT_SLAVE: 0 = Use slave addr0. 1 = Use slave addr1.
10:9	0x0	SL1_ADDR_HI: In 7-bit address mode, these bits are don't care. In 10-bit address mode, they represent the 2 MSBs of the address.
8	0x0	SL1_VLD: 0 = 7-bit addressing. 0 = SEVEN_BIT_ADDR_MODE 1 = TEN_BIT_ADDR_MODE
2:1	0x0	SL_ADDR_HI: In 7-bit address mode, these bits are don't care. In 10-bit address mode, they represent the 2 MSBs of the address.
0	0x0	VLD: 0 = 7-bit addressing. 1 - 10 bit addressing. 0 = SEVEN_BIT_ADDR_MODE 1 = TEN_BIT_ADDR_MODE

35.5.12 I2C_I2C_TLOW_SEXT_0

I2C Controller SMBUS Timeout Thresholds

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx00000000000000000000000000)

Bit	Reset	Description
27	0x0	RST_SL_ON_TIMEOUT: Reset Slave state machine on time-out
26	0x0	TLOW_MEXT_EN: Enable TLOW_MEXT counter
25	0x0	TLOW_SEXT_EN: Enable TLOW_SEXT counter
24	0x0	TIMEOUT_EN: Enable TIMEOUT counter
23:16	0x0	TLOW_MEXT: Cumulative clock low extend time (master device) accumulated over a byte transfer period in milliseconds (START to ACK, ACK to ACK, or ACK to STOP).
15:8	0x0	TLOW_SEXT: Cumulative clock low extend time (slave device) accumulated over a complete transfer (START till STOP)
7:0	0x0	TIMEOUT: Clock low time-out period in milliseconds

35.5.13 I2C_I2C_SL_DELAY_COUNT_0

I2C Slave Controller Delay Count

Offset: 0x3c | Read/Write: R/W | Reset: 0x0000001e (0bxxxxxxxxxxxxxxxx0000000000011110)

Bit	Reset	Description
15:0	0x1e	SL_DELAY_COUNT: The value determines the timing between an address cycle and a subsequent data cycle or two consecutive data cycles on the bus. The I2C_SL_DELAY_COUNT is valid only when an internal slave is accessed. I2C_SL_DELAY_COUNT has to be programmed such that TIMING = T * DLY, where T is the period of the clock source selected for I2C, DLY is I2C_SL_DELAY_COUNT, and TIMING is the desired timing. A value of >= 1250 ns is recommended.

35.5.14 I2C_I2C_SL_INT_MASK_0

I2C Controller Slave Mask (Slave)

Offset: 0x40 | Read/Write: R/W | Reset: 0x000000fd (0bxxxxxxxxxxxxxxxxxxxxxxxxx111111x1)

Bit	Reset	Description
7	0x1	HW_MSTR_INT: 0 = DISABLE 1 = ENABLE
6	0x1	REPROG_SL: 0 = DISABLE 1 = ENABLE
5	0x1	RST_SL: 0 = DISABLE 1 = ENABLE
4	0x1	END_TRANS: 0 = DISABLE 1 = ENABLE
3	0x1	SL_IRQ: 0 = DISABLE 1 = ENABLE
2	0x1	RCVD: 0 = DISABLE 1 = ENABLE
0	0x1	ZA: 0 = DISABLE 1 = ENABLE

35.5.15 I2C_I2C_SL_INT_SOURCE_0

I2C Controller Slave Source (Slave)

Offset: 0x44 | Read/Write: RO | Reset: 0x000000XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
7	X	HW_MSTR_INT: 0 = UNSET 1 = SET
6	X	REPROG_SL: 0 = UNSET 1 = SET
5	X	RST_SL: 0 = UNSET 1 = SET
4	X	END_TRANS: 0 = UNSET 1 = SET
3	X	SL_IRQ: 0 = UNSET 1 = SET
2	X	RCVD: 0 = UNSET 1 = SET

Bit	Reset	Description
0	X	ZA: 0 = UNSET 1 = SET

35.5.16 I2C_I2C_SL_INT_SET_0

I2C Controller Slave Source (Slave)

Offset: 0x48 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx000000x0)

Bit	Reset	Description
7	0x0	HW_MSTR_INT: 0 = UNSET 1 = SET
6	0x0	REPROG_SL: 0 = UNSET 1 = SET
5	0x0	RST_SL: 0 = UNSET 1 = SET
4	0x0	END_TRANS: 0 = UNSET 1 = SET
3	0x0	SL_IRQ: 0 = UNSET 1 = SET
2	0x0	RCVD: 0 = UNSET 1 = SET
0	0x0	ZA: 0 = UNSET 1 = SET

Note: Program the field PACKET_MODE_EN of I2C_CNFG register while working in packet mode.

The set of registers given below describe the interface for packet mode only. With packet mode interface changes, normal mode registers and the operation are not affected. The transition from normal mode to packet mode is suggested because packet mode allows:

- There is no restriction on the number of bytes before and the after the repeated start
- The number of bytes that can be transferred with a single command is not limited to 8.

Though a packet can contain 4 Kbytes, because any number of packets can be pushed into the FIFO, there is no limit on the number of bytes that can be transferred.

- The transactions to different slaves can be chained together with repeat start and there is no limit on the number of slaves it can address.

35.5.17 I2C_I2C_TX_PACKET_FIFO_0

A packet contains header and payload. The header size is variable and could vary from 2 to 5 words. For I2C, the header size is 3 words. The first two words of the header contain generic information. The third word contains I2C transaction-specific information. The payload contains actual data to be written to the slave. For read operations, the payload is nil. Thus, the packet contains a header only.

Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_PACKET: Software writes packets into this register. A packet may contain generic information

35.5.18 I2C_I2C_RX_FIFO_0

Header or I2C Specific Header or Data

Offset: 0x54 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RD_DATA: Software reads data from this register, causes a pop

35.5.19 I2C_PACKET_TRANSFER_STATUS_0

Offset: 0x58 | Read/Write: RO | Reset: 0x0XXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
24	X	TRANSFER_COMPLETE: The packet transfer for which the last packet is set has been completed 0 = UNSET 1 = SET
23:16	X	TRANSFER_PKT_ID: The current packet ID for which the transaction is happening on the bus
15:4	X	TRANSFER_BYTENUM: The number of bytes transferred in the current packet
3	X	NOACK_FOR_ADDR: No ACK received for the address byte 0 = UNSET 1 = SET
2	X	NOACK_FOR_DATA: No ACK received for the data byte 0 = UNSET 1 = SET
1	X	ARB_LOST: Arbitration lost for the current byte 0 = UNSET 1 = SET
0	X	CONTROLLER_BUSY:1 = Controller is busy 0 = UNSET 1 = SET

35.5.20 I2C_FIFO_CONTROL_0

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:13	0x0	SLV_TX_FIFO_TRIG: Slave Transmit FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word is empty in the FIFO 001 = 2 word DMA trigger is asserted when at least 2 words are empty in the FIFO

Bit	Reset	Description
12:10	0x0	SLV_RX_FIFO_TRIG: Slave Receive FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word is full in the FIFO. 001 = 2 word DMA trigger is asserted when at least 2 words are full in the FIFO
9	0x0	SLV_TX_FIFO_FLUSH:1 = Flush the TX FIFO. Cleared after the FIFO is flushed 0 = UNSET 1 = SET
8	0x0	SLV_RX_FIFO_FLUSH:1 = Flush the RX FIFO. Cleared after the FIFO is flushed 0 = UNSET 1 = SET
7:5	0x0	TX_FIFO_TRIG: Transmit FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word is empty in the FIFO. 001 = 2 word DMA trigger is asserted when at least 2 words are empty in the FIFO
4:2	0x0	RX_FIFO_TRIG: Receive FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word is full in the FIFO. 001 = 2 word DMA trigger is asserted when at least 2 words are full in the FIFO
1	0x0	TX_FIFO_FLUSH:1 = Flush the TX FIFO. Cleared after the FIFO is flushed. 0 = UNSET 1 = SET
0	0x0	RX_FIFO_FLUSH:1 = Flush the RX FIFO. Cleared after the FIFO is flushed. 0 = UNSET 1 = SET

35.5.21 I2C_FIFO_STATUS_0

Offset: 0x60 | Read/Write: RO | Reset: 0x0XXX00XX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
25	X	SLV_XFER_ERR_REASON: This bit describes the nature of the packet transfer error. It is meaningful only if PKT_XFER_ERR is set. 0 = The Master terminated the transaction before it was completed. 1 = The Master did not terminate the transaction when all bytes are transferred.
23:20	X	SLV_TX_FIFO_EMPTY_CNT: The number of slots that can be written to the slave TX FIFO. 0000 = tx_fifo full. 0001 = 1 slot empty. 0010 = 2 slots empty
19:16	X	SLV_RX_FIFO_FULL_CNT: The number of slots to be read from the Slave RX FIFO. 0000 = rx_fifo empty. 0001 = 1 slot full. 0010 = 2 slots full
7:4	X	TX_FIFO_EMPTY_CNT: The number of slots that can be written to the TX FIFO. 0000 = tx_fifo full. 0001 = 1 slot empty. 0010 = 2 slots empty
3:0	X	RX_FIFO_FULL_CNT: The number of slots to be read from the RX FIFO. 0000 = rx_fifo empty. 0001 = 1 slot full. 0010 = 2 slots full

35.5.22 I2C_INTERRUPT_MASK_REGISTER_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0bxxx000000000xx00xxxx000000000000)

Bit	Reset	Description
28	0x0	SLV_ACK_WITHHELD_INT_EN: 0 = DISABLE 1 = ENABLE
27	0x0	SLV_RD2WR_INT_EN: 0 = DISABLE 1 = ENABLE
26	0x0	SLV_WR2RD_INT_EN: 0 = DISABLE 1 = ENABLE
25	0x0	SLV_PKT_XFER_ERR_INT_EN: 0 = DISABLE 1 = ENABLE
24	0x0	SLV_TX_BUFFER_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
23	0x0	SLV_RX_BUFFER_FILLED_INT_EN: 0 = DISABLE 1 = ENABLE
22	0x0	SLV_PACKET_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
21	0x0	SLV_TFIFO_OVF_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
20	0x0	SLV_RFIFO_UNF_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
17	0x0	SLV_TFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
16	0x0	SLV_RFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
11	0x0	BUS_CLEAR_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
10	0x0	TLOW_MEXT_TIMEOUT_EN: Used as Slave Timeout interrupt enable 0 = DISABLE 1 = ENABLE
9	0x0	TLOW_SEXT_TIMEOUT_EN: Used as Master Timeout interrupt enable 0 = DISABLE 1 = ENABLE
8	0x0	TIMEOUT_INT_EN: 0 = DISABLE 1 = ENABLE
7	0x0	PACKET_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
6	0x0	ALL_PACKETS_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	TFIFO_OVF_INT_EN: 0 = DISABLE 1 = ENABLE
4	0x0	RFIFO_UNF_INT_EN: 0 = DISABLE 1 = ENABLE
3	0x0	NOACK_INT_EN: 0 = DISABLE 1 = ENABLE
2	0x0	ARB_LOST_INT_EN: 0 = DISABLE 1 = ENABLE
1	0x0	TFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
0	0x0	RFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE

35.5.23 I2C_INTERRUPT_STATUS_REGISTER_0

This register indicates the status bit for which the interrupt is set. If the interrupt is set, write a 1 to clear it. However, the TFIFO_DATA_REQ and RFIFO_DATA_REQ fields depend on the FIFO trigger levels and cannot be cleared.

Slv_rd2wr indicates there is a switch from read to write by repeat start and the current read transaction needs to be closed and started with a write transaction. Similarly, slv_wr2rd indicates a switch from write to read.

Offset: 0x68 | Read/Write: R/W | Reset: 0x000X000X (0bxxx000000000xxxxxxxx0000000000xx)

Bit	R/W	Reset	Description
28	RW	0x0	SLV_ACK_WITHHELD: ACK is withheld, waiting for software explicit information about the ACK 0 = UNSET 1 = SET
27	RW	0x0	SLV_RD2WR:Transaction switching from read to write 0 = UNSET 1 = SET
26	RW	0x0	SLV_WR2RD:Transaction switching from write to read 0 = UNSET 1 = SET
25	RW	0x0	SLV_PKT_XFER_ERR: 0 = Request was successful. 1 = Error has occurred during packet transfer 0 = UNSET 1 = SET
24	RW	0x0	SLV_TX_BUFFER_REQ: Slave TX buffer is full 0 = UNSET 1 = SET
23	RW	0x0	SLV_RX_BUFFER_FILLED: Slave RX buffer is full 0 = UNSET 1 = SET
22	RW	0x0	SLV_PACKET_XFER_COMPLETE: Slave packet transfer complete 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
21	RW	0x0	SLV_TFIFO_OVF: Slave TX FIFO overflow 0 = UNSET 1 = SET
20	RW	0x0	SLV_RFIFO_UNF: Slave RX FIFO underflow 0 = UNSET 1 = SET
17	RO	X	SLV_TFIFO_DATA_REQ: Slave TX FIFO data request 0 = UNSET 1 = SET
16	RO	X	SLV_RFIFO_DATA_REQ: Slave RX FIFO data request 0 = UNSET 1 = SET
11	RW	0x0	BUS_CLEAR_DONE: Bus clear done status 0 = UNSET 1 = SET
10	RW	0x0	TLOW_MEXT_TIMEOUT: SMBUS mext time-out 0 = UNSET 1 = SET
9	RW	0x0	TLOW_SEXT_TIMEOUT:SMBUS sext time-out 0 = UNSET 1 = SET
8	RW	0x0	TIMEOUT:SMBUS time-out 0 = UNSET 1 = SET
7	RW	0x0	PACKET_XFER_COMPLETE: A packet has been transferred successfully. The TRANSFER_PKT_ID field can be used to know the current byte under transfer. This bit can be masked by the IE field in the I2C specific header. 0 = UNSET 1 = SET
6	RW	0x0	ALL_PACKETS_XFER_COMPLETE: All packets transferred successfully 0 = UNSET 1 = SET
5	RW	0x0	TFIFO_OVF: TX FIFO overflow 0 = UNSET 1 = SET
4	RW	0x0	RFIFO_UNF: RX FIFO underflow 0 = UNSET 1 = SET
3	RW	0x0	NOACK: No ACK from slave 0 = UNSET 1 = SET
2	RW	0x0	ARB_LOST: Arbitration lost 0 = UNSET 1 = SET
1	RO	X	TFIFO_DATA_REQ: TX FIFO data request 0 = UNSET 1 = SET
0	RO	X	RFIFO_DATA_REQ: RX FIFO data request 0 = UNSET 1 = SET

35.5.24 I2C_I2C_CLK_DIVISOR_REGISTER_0

The divisor values (N) must be programmed so that:

- SCL frequency (Standard/Fast mode) = ClkSourceFreq/(8*(N+1))
- SCL frequency (High Speed mode) = ClkSourceFreq/(12*(N+1))

Offset: 0x6c | Read/Write: R/W | Reset: 0x00190001 (0b00000000000110010000000000000001)

Bit	Reset	Description
31:16	0x19	I2C_CLK_DIVISOR_STD_FAST_MODE: N = divide by n+1
15:0	0x1	I2C_CLK_DIVISOR_HSMODE: N = divide by n+1

35.5.25 I2C_I2C_INTERRUPT_SOURCE_REGISTER_0

This read-only register returns the AND of the Interrupt Source and Interrupt Mask registers.

Offset: 0x70 | Read/Write: RO | Reset: 0xXXXX0XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28	X	SLV_ACK_WITHHELD: 0 = UNSET 1 = SET
27	X	SLV_RD2WR: 0 = UNSET 1 = SET
26	X	SLV_WR2RD: 0 = UNSET 1 = SET
25	X	SLV_PKT_XFER_ERR: Error occurred during a slave transfer 0 = UNSET 1 = SET
24	X	SLV_TX_BUFFER_REQ: Slave TX buffer is full 0 = UNSET 1 = SET
23	X	SLV_RX_BUFFER_FILLED: Slave RX buffer is full 0 = UNSET 1 = SET
22	X	SLV_PACKET_XFER_COMPLETE: Slave packet transfer complete 0 = UNSET 1 = SET
21	X	SLV_TFIFO_OVF: Slave TX FIFO overflow 0 = UNSET 1 = SET
20	X	SLV_RFIFO_UNF: Slave RX FIFO underflow 0 = UNSET 1 = SET
17	X	SLV_TFIFO_DATA_REQ: Slave TX FIFO data request 0 = UNSET 1 = SET
16	X	SLV_RFIFO_DATA_REQ: Slave RX FIFO data request 0 = UNSET 1 = SET
11	X	BUS_CLEAR_DONE: Bus clear done 0 = UNSET 1 = SET

Bit	Reset	Description
10	X	TLOW_MEXT_TIMEOUT: SMBUS mext time-out 0 = UNSET 1 = SET
9	X	TLOW_SEXT_TIMEOUT: SMBUS sext time-out 0 = UNSET 1 = SET
8	X	TIMEOUT: SMBUS time-out 0 = UNSET 1 = SET
7	X	PACKET_XFER_COMPLETE: Packet transferred successfully 0 = UNSET 1 = SET
6	X	ALL_PACKETS_XFER_COMPLETE: All the packets transferred successfully 0 = UNSET 1 = SET
5	X	TFIFO_OVF: TX FIFO overflow 0 = UNSET 1 = SET
4	X	RFIFO_UNF: RX FIFO underflow 0 = UNSET 1 = SET
3	X	NOACK: No ACK from slave 0 = UNSET 1 = SET
2	X	ARB_LOST: Arbitration lost 0 = UNSET 1 = SET
1	X	TFIFO_DATA_REQ: TX FIFO data request 0 = UNSET 1 = SET
0	X	RFIFO_DATA_REQ: RX FIFO data request 0 = UNSET 1 = SET

35.5.26 I2C_I2C_INTERRUPT_SET_REGISTER_0

This write-only register can be used to set the interrupt status bit. A write to this register causes the bits in the status register to be set if the corresponding bit in the write data is 1'b1. A read always returns 'h0.

Offset: 0x74 | Read/Write: R/W | Reset: 0x00000000 (0bxxx000000000xxxxxxxx0000000000xx)

Bit	Reset	Description
28	0x0	SLV_ACK_WITHHELD: 0 = UNSET 1 = SET
27	0x0	SLV_RD2WR: 0 = UNSET 1 = SET
26	0x0	SLV_WR2RD: 0 = UNSET 1 = SET
25	0x0	SLV_PKT_XFER_ERR: Error occurred during slave transfer 0 = UNSET 1 = SET

Bit	Reset	Description
24	0x0	SLV_TX_BUFFER_REQ: Slave TX buffer is full 0 = UNSET 1 = SET
23	0x0	SLV_RX_BUFFER_FILLED: Slave RX buffer is full 0 = UNSET 1 = SET
22	0x0	SLV_PACKET_XFER_COMPLETE: Slave packet transfer complete 0 = UNSET 1 = SET
21	0x0	SLV_TFIFO_OVF: Slave TX FIFO overflow 0 = UNSET 1 = SET
20	0x0	SLV_RFIFO_UNF: Slave RX FIFO underflow 0 = UNSET 1 = SET
11	0x0	BUS_CLEAR_DONE: Bus clear done 0 = UNSET 1 = SET
10	0x0	TLOW_MEXT_TIMEOUT:SMBUS mext time-out 0 = UNSET 1 = SET
9	0x0	TLOW_SEXT_TIMEOUT:SMBUS sext time-out 0 = UNSET 1 = SET
8	0x0	TIMEOUT:SMBUS time-out 0 = UNSET 1 = SET
7	0x0	PACKET_XFER_COMPLETE: Packet transferred successfully 0 = UNSET 1 = SET
6	0x0	ALL_PACKETS_XFER_COMPLETE: All the packets transferred successfully 0 = UNSET 1 = SET
5	0x0	TFIFO_OVF: TX FIFO overflow 0 = UNSET 1 = SET
4	0x0	RFIFO_UNF: RX FIFO underflow 0 = UNSET 1 = SET
3	0x0	NOACK: No ACK from slave 0 = UNSET 1 = SET
2	0x0	ARB_LOST: Arbitration lost 0 = UNSET 1 = SET

35.5.27 I2C_I2C_SLV_TX_PACKET_FIFO_0

Offset: 0x78 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TX_PACKET: Software writes packets into this register. A packet may contain generic information.

35.5.28 I2C_I2C_SLV_RX_FIFO_0

Header or I2C Specific Header or Data

Offset: 0x7c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RD_DATA: Software reads data from this register, causes a pop.

35.5.29 I2C_I2C_SLV_PACKET_STATUS_0

Offset: 0x80 | Read/Write: RO | Reset: 0x0XXXXXX0 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
25	X	ACK_WITHHELD: Indicates that an ACK is withheld for the last byte and the slave is waiting for the host to explicitly command the slave to acknowledge the last byte. 0 = Bus is released. 1 = ACK is withheld
24	X	TRANSFER_COMPLETE: All the packets have been transferred successfully
23:16	X	TRANSFER_PKT_ID: The current packet ID for which the transaction is happening on the bus
15:4	X	TRANSFER_BYTENUM: The number of bytes transferred in the current packet

35.5.30 I2C_I2C_BUS_CLEAR_CONFIG_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00090004 (0bxxxxxxxx00001001xxxxxxxxxxxxx100)

Bit	Reset	Description
23:16	0x9	BC_SCLK_THRESHOLD: send the clock pulses until this threshold is met
2	0x1	BC_STOP_COND: 0 = NO_STOP : Do not send a stop condition at the end of the bus clear operation 1 = STOP : Send a stop condition at the end of the bus clear operation
1	0x0	BC_TERMINATE: 0 = IMMEDIATE: Terminate the bus clear operation immediately when SDA is released or threshold count is reached, whichever is earlier 1 = THRESHOLD: Irrespective of SDA release status during bus clear, terminate the bus clear only after the threshold is reached.
0	0x0	BC_ENABLE: Starts bus clear operation. Hardware auto-clears this bit upon bus clear transaction completion

35.5.31 I2C_I2C_BUS_CLEAR_STATUS_0

Offset: 0x88 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
0	X	BC_STATUS: 0 = NOT_CLEARED: Indicates SDA is not released by the slave. Its status is still low. 1 = CLEARED: SDA is released

35.5.32 I2C_I2C_SPARE_0

Spare Register

Offset: 0x8c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	Reserved

36.0 UART AND VFIR CONTROLLER

There are four Universal Asynchronous Receiver/Transmitters (UARTs) built into the Tegra® 4 devices. These UARTs support both 16450 and 16550 compatible modes. VFIR functionality is also supported.

36.1 Functional Description

36.1.1 UARTs A through D

All UARTs provide serial data synchronization and data conversion (parallel-to-serial and serial-to-parallel) for both receiver and transmitter sections. Synchronization for serial data stream is accomplished by adding start and stop bits to the transmit data to form a data character. Data integrity is accomplished by attaching a parity bit to the data character. The parity bit can be checked by the receiver for any transmission bit errors.

The COM-HOST interface is fully programmable through an 8-bit CPU interface. The registers are 32-bit word aligned. The interface supports word lengths from five to eight bits, an optional parity bit, and one or two stop bits. If enabled, parity can be odd, even, or forced to a defined state. Interrupts can be generated from any of 10 sources.

The UARTs support both 16450 and 16550 compatible modes. The default mode is 16450. This mode provides independent 16-byte FIFOs for transmit and receive operations and is selected by the FIFO control register. It also includes a 16-bit programmable baud rate generator and an 8-bit scratch register, 8 modem control lines, and 2 DMA handshake lines that are used to indicate when the FIFOs are ready to transfer data to the CPU.

The UARTs support a device clock of up to 72 MHz. The maximum baud rate is 16 clock cycles per symbol or 4.5 Megabits per second.

UART B is identical to UART A with the exception that only two signals are connected to its pins. UART A, C, and D are identical.

36.1.2 VFIR Controller

The VFIR controller in the Tegra 4 device implements the control and data sections of the IrDA Physical layer version 1.4, handling handshaking and basic networking between devices, and interfaces to an external Infrared transceiver.

This controller implements three distinct speed ranges and protocols: 2400 to 115,200 bits per second (Serial IR or SIR), 4 Megabits per second (called Fast Infrared or FIR), and 16 Megabits per second (called Very Fast Infrared or VFIR). Initial communication starts at 9600 bits per second, and negotiations proceed either faster or slower as the link strength allows. Each of the three speed ranges has different communication protocols and encoding schemes.

The following table lists the IrDA supported protocols.

Table 114: Protocols Supported by the IrDA

Signaling Rate (bps)	IrDA Name	Frame Wrapper and Encoding	IrDA Status	Tegra 4 Device Support?
9.6 K	SIR	Async RZI (3/16)	Mandatory	Yes
2400 - 115.2K	SIR	Async RZI (3/16)	Recommended	Yes
576.0 K	MIR	Sync HDLC RZI (4/16)	Optional	No
1.152 M	MIR	Sync HDLC RZI (4/16)	Optional	No
4.0 M	FIR	Sync 4PPM	Optional	Yes

Signaling Rate (bps)	IrDA Name	Frame Wrapper and Encoding	IrDA Status	Tegra 4 Device Support?
16.0 M	VFIR	Sync HHH(1,13)	Optional	Yes

The VFIR controller can operate in three different encodings; each is implemented separately. In SIR mode, a pulse encoder/decoder is inserted in the transmit/receive path of a standard UART. FIR and VFIR modes have their own control blocks. The outputs of the three blocks are multiplexed based on the current operating mode (SIR, FIR, or VFIR), before they leave the Tegra 4 device processor. The input signals are demultiplexed in a similar fashion.

Note: The operating mode is selected with the Mode [2:0] bits of the VFIR.CTL register

Figure 144: UART Block Diagram with SIR Decoder and Encoder

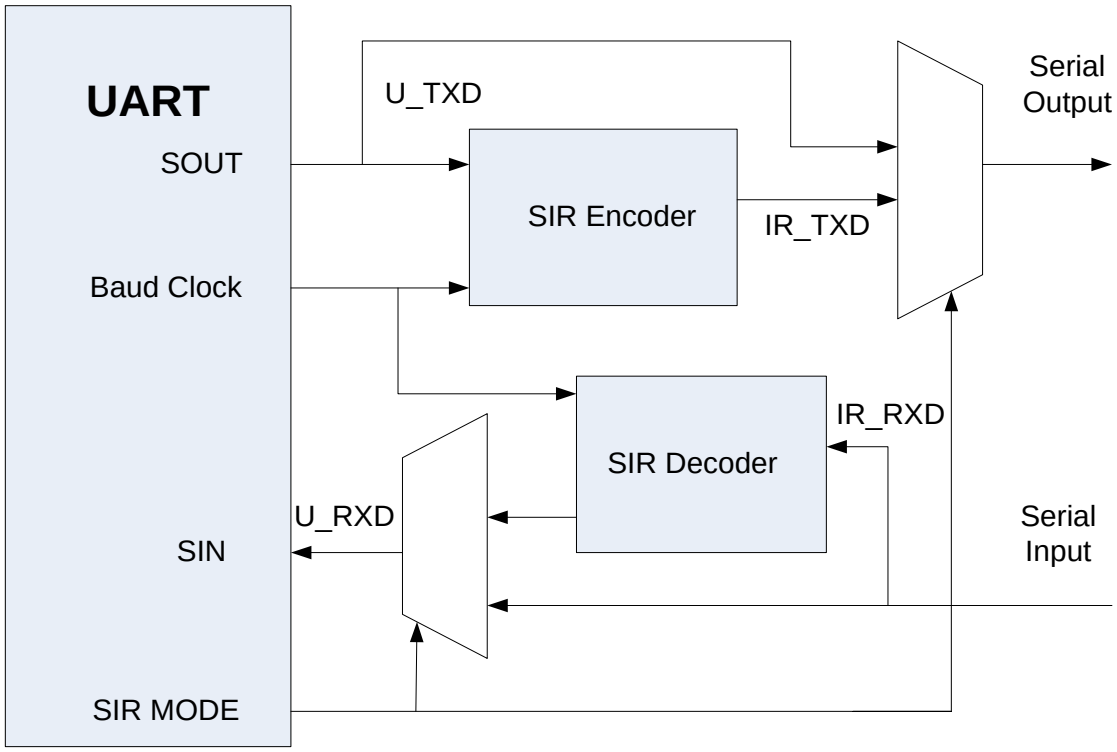
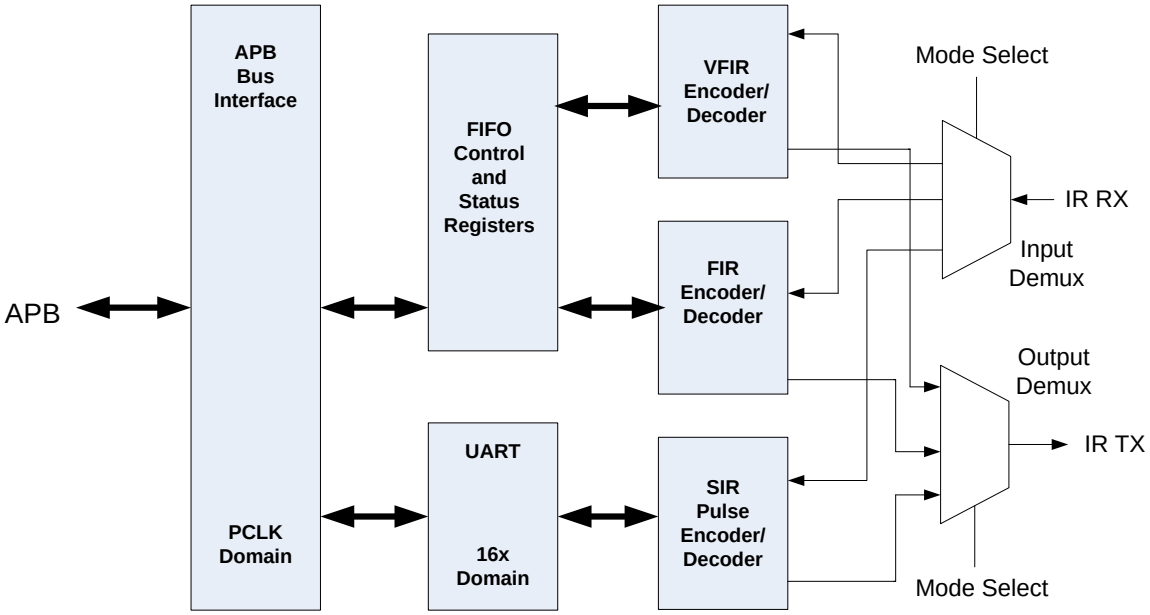


Figure 145: VFIR Controller Top-Level Block Diagram



36.1.3 Hardware Features

The features supported by UART A, UART B, UART C, and UART D are:

- Synchronization for the serial data stream with start and stop bits to transmit data to and from a data character
- Data integrity by attaching a parity bit to the data character
- The COM-HOST interface is fully programmable through an 8-bit CPU interface
- Support for word lengths from 5 to 8 bits, an optional parity bit, and 1 or 2 stop bits
- The UART supports both 16450- and 16550-compatible modes. The default mode is 16450.
- DMA capable for both TX and RX
- 8 bit x 16 deep FIFO
- Auto sense baud detection
- Timeout interrupts to indicate if the incoming stream stopped
- Flow control support on RTS and CTS

The features supported by the VFIR controller are:

- Supports IrDA ver1.0 SIR protocol with maximum baud rate to 115.2 Kilobits per second
- Supports IrDA ver1.3 FIR protocol (4 Megabits per second)
- Supports IrDA ver1.4 VFIR protocol (16 Megabits per second)
- Programmable polarities on all the interface pins
- DMA capable for both TX and RX
- 32 bit x 16 deep FIFO
- Diagnostics for loopback and error insertion
- Interface control, such as transceiver signal strength, must be performed through GPIOs
- The maximum frequency of the device clock is 72 MHz

36.1.4 Hardware Signaling

All UARTs in the Tegra 4 device are implemented by a hardware block that supports modem control signals such as DTR, DSR, and DCD. UART A is, however, the only UART that allows those signals to be used externally.

For UART A, when the pin-mux is configured for 4- or 2-pin UART operation, the DSR and DCD input signals are tied low (active).

For UART B, the DTR output is routed back into the DSR and DCD inputs in hardware. This means that if software changes DTR state, the UART may raise an interrupt due to DSR/DCD changing state.

For UARTs C and D, the DSR and DCD inputs are tied to an inactive value.

Table 115: Serial Bus Interface Signals

Signal Name	Description
Outputs	
TXD	Transmit Data Port
RTS	Request to Send
DTR	Data Terminal Ready
Inputs	
RXD	Receiver Data Port
CTS	Clear to Send
DSR	Data Set Ready
DCD	Data Carrier Detect
RI	Ring Indicator

Note: The DSR, DCD and RI MODEM control signals do not control any hardware other than generating interrupts and status on change.

36.2 UART Programming Guidelines

36.2.1 16450 Mode Programming

1. Program pin mux settings to select a UART
2. Enable the UART clocks
3. For enabling internal loopback, program MCR[4] to 1
4. Program FCR[0] to 0
5. Enable interrupts in the IER register as needed
6. Write data into the THR register
7. Wait for a THR interrupt, if enabled, or poll for LSR[5]
8. During a receive, wait for an RBR interrupt or poll for LSR[0]
9. Read the UART.LSR register to clear interrupts

36.2.2 16550 Mode Programming

1. Program pin mux settings to select a UART
2. Enable the UART clocks
3. For enabling internal loopback, program MCR[4] to 1
4. Program FCR[0] to 1
5. Program trigger levels as required
6. Enable interrupts in the IER register as needed
7. Write data into the THR register
8. Wait for a THR interrupt, if enabled, or poll for LSR[5]
9. During a receive, wait for an RBR interrupt or poll for LSR[0]
10. Read the UART.LSR register to clear interrupts
11. The APB-DMA requestor numbers for UARTs A, B, and C are 8, 9, and 10, respectively

36.2.3 Transmitter and Receiver Holding Registers

The serial transmitter section consists of a Transmit Hold Register (THR) and Transmit Shift Register (TSR). The status of transmit hold register is provided in the Line Status Register (LSR). Writing to this register (THR) transfers the contents of data bus (D [7:0]) to the Transmit Holding Register whenever the Transmitter Holding Register or Transmitter Shift Register is empty. The Transmit Holding Register empty flag is set to 1 when the transmitter is empty or data is transferred to the transmit shift register. Note that a write operation is performed when the Transmit Holding Register empty flag is set.

The serial receiver section also contains an 8-bit Receiver Holding/Buffer Register (RBR). Receive data is removed from the UART and received by the processor by reading the RBR. The receiver contains a mechanism for preventing false starts as follows: On the falling edge of the start bit, the receiver internal counter starts to count 7.5 clocks (16x clock), which is the center of the start bit. The start bit is valid if the RX is still low at the mid-bit sample of the start bit. Verifying the start bit prevents the receiver from assembling a false data character due to a low going noise spike on the RX input. Receiver status codes are posted in the Line Status Register.

36.2.4 FIFO Interrupt Mode Operation

When the receive FIFO (FCR bit 0 = 1) and receive interrupts (IER bit 0 = 1) are enabled, a receiver interrupt occurs as follows:

- The receive data available interrupts are issued to the CPU when the FIFO has reached its programmed trigger level; it is cleared as soon as the FIFO drops below its programmed trigger level.
- The ISR receive data available indication also occurs when the FIFO trigger level is reached, and like the interrupt, it is cleared when the FIFO drops below the trigger level.
- The data ready bit (LSR bit 0) is set as soon as a character is transferred from the shift register to the receiver FIFO. It is reset when the FIFO is empty.

36.2.5 FIFO Polled Mode Operation

When FCR bit 0 = 1; clearing IER bits [3:0] to zero puts the UART in the FIFO polled mode of operation. Since the receiver and transmitter are controlled separately, either one or both can be in the polled mode operation by using the line status register as follows:

- LSR bit 0 is set as long as there is one byte in the receive FIFO
- LST bits [4:1] specifies which error(s) have occurred
- LSR bit 5 indicates when the transmit FIFO is empty
- LSR bit 6 indicates when both transmit FIFO and transmit shift registers are empty
- LSR bit 7 indicates when there are any errors in the receive FIFO

The UART requires a two-step FIFO enable operation in order to enable receive trigger levels.

36.2.6 Programmable Baud Rate Generator

The UART contains a programmable baud rate generator that can take the UART clock input and divide it by any divisor from 1 to 2^{16}-1. Refer to the Clocking and Reset Controller section for definitions of the UART clocks. The output frequency of baudout is equal to 16X the transmission baud rate (Baudout=16 X baud rate). Customized baud rates are achieved by selecting proper divisor values for the MSB and LSB bits of the baud rate generator.

36.2.7 Enable Register (IER)

There is an Interrupt Enable Register for each UART (UART A Interrupt Enable and UART B Interrupt Enable). The Interrupt Enable Register(s) masks the incoming interrupts from the receiver ready, transmitter empty, line status, and modem status registers to the INT output pin.

36.2.8 Interrupt Identification Register (IIR)

The UART provides four level prioritized interrupt conditions to minimize software overhead during data character transfers. The Interrupt Identification Register (IIR) provides the source of the interrupt in a prioritized manner.

During the read cycle, the 16550 provides the highest interrupt level to be serviced by the CPU. No other interrupts are acknowledged until the particular interrupt is serviced. The prioritized interrupt levels are shown in the following table. The Receive Data Time-out mode is enabled when the UART is operating in the FIFO mode.

Receive time-out does not occur if receive FIFO is empty. The time-out counter is reset at the center of each stop bit received or each time the Receive Holding Register is read. The actual time out value is:

- T (Time out length in bits) = 4 X P (Programmed work length) + 12

To convert the time-out value to a character value, divide this number to its complete word length + parity (if used) + number of stop bits and start bit.

Example

If you program the word length = 7 with no parity and one stop bit, the time-out is:

- T = 4X7 (programmed word length) + 12 = 40 bits.
- Character time = 40/9
- [(Programmed word length = 7) + (stop bit = 1) + (start bit = 1)] =4.4 characters

Table 116 Prioritized Interrupt Levels

Priority	D3	D2	D1	D0	Interrupt Source Description
1	0	1	1	0	LSR (Receiver Line Status Register)
2	0	1	0	0	RXRDY (Received Data Ready)
2	1	1	0	0	RXRDY (Received Data Timeout)
3	0	0	1	0	TXRDY (Transmitter Holding Register)
4	0	0	0	0	MSR (Modem Status Register)

36.2.9 FIFO Control Register (FCR) Modes

The FIFO control register is used to enable the FIFOs, clear the FIFOs, set the receiver FIFO trigger level, and select the type of DMA signaling. The operation of the FCR in the four DMA modes is given below.

36.2.9.1 Transmit Operation in DMA Mode 0 or Mode 1

When the UART is in the 16450 mode (FCR bit 0 = 0) or in the FIFO mode (FCR bit 0 = 1, FCR bit 3 = 0) and when there are no characters in the transmit FIFO or transmit holding register, the TXRDY* pin goes low. Once active, the TXRDY* pin goes high (inactive) after the first character is loaded into the Transmit Holding Register.

36.2.9.2 Receive Operation in DMA Mode 0

When the UART is in 16450 mode (FCR bit 0 = 0) or in the FIFO mode (FCR bit 0 = 1,FCR bit 3 = 0) and there is at least 1 character in the receive FIFO, the RXRDY* pin goes low. Once active, the RXRDY* pin goes high (inactive) when there are no more characters in the receiver.

36.2.9.3 Receive Operation in DMA Mode 1

When the UART is in the FIFO mode (FCR bit 0 = 1, FCR bit 3 = 1) and the trigger level or the timeout has been reached, the RXRDY* pin goes low. Once it is activated, it goes high (inactive) when there are no more characters in the FIFO.

36.2.10 Line Control Register (LCR)

The Line Control Register is used to specify the asynchronous data communication format. The word length, stop bits, and parity can be selected by writing appropriate bits in this register.

36.2.11 Modem Control Register (MCR)

This register controls the interface with the modem or a peripheral device (RS232).

Loopback Mode

If MCR[4] = 1, the loopback mode is enabled, and the following occurs:

The transmitter output (TX) is set high (Mark condition), and the receiver input (RX), CTS, DSR, CD, and RI are disabled. Internally, the transmitter output is connected to the receiver input and DTR, RTS, OP1, and OP2 are connected to modem control inputs.

In this mode, the receiver and transmitter interrupts are fully operational, but the interrupt sources are now the lower four bits of the modem control register instead of the four modem control inputs. The interrupts are controlled by the IER register.

Table 117: Modem Control Register (MCR)

Bit	Name	Description
7	N/A	Not used. Set to 0 internally.
6	RTS_EN	1 = Enable Hardware Flow Control using RTS 0 = Disable Hardware Flow Control
5	CTS_EN	1 = Enable Hardware Flow Control using CTS 0 = Disable Hardware Flow Control
4	LOOP	0 = Normal operating mode. 1 = Enable local loop-back mode (diagnostic).
3	OUT2	nOUT2 polarity
2	OUT1	nOUT1 polarity
1	RTS	1 = Force RTS (Request to Send) low 0 = Force RTS to high
0	DTR	1 = Force DTR (Data Terminal Ready) to low 1 = Force to high

36.2.12 Line Status Register (LSR)

The Line Status Register provides the status of data transfer to the CPU.

36.2.13 Modem Status Register (MSR)

The modem status register provides the current state of the control lines from the modem or peripheral to the CPU. Four bits of this register are used to indicate the changed information. These bits are set to 1 whenever a control input from the modem changes state. They are set to 0 whenever the CPU reads the register. The following table shows the bit format for the MSR register.

36.2.14 Scratchpad Register (SR)

Eight bits of information can be stored in this register. Information in this register does not affect the operation of the device in any way.

36.2.15 Autobaud Sense Register (ASR)

The UART can automatically determine the correct baud divisor by using the Autobaud Sense Register (UART offsets 0x3C). The most significant bit of this register is the valid flag. A write to this register will clear the valid flag and enable the autobaud process. When the first RX edge occurs, a counter running at 24 MHz starts counting. When another rx_edge occurs, the "complete" flag is set, the value is frozen, and the Autobaud Sense Value register is updated with the count value. The low 20 bits of the ASR give the number of clocks within a single bit. Because the UART uses 16x oversampling, the resulting value needs to be adjusted by shifting right 4 bits, then loading the resulting count in the divisor latch of the UART. (In the code snippet below, the lower 4 bits are rounded to give slightly greater accuracy.)

Because the speed determination is made by measuring the start bit, special characters must be sent by the transmitting UART to guarantee that the next character after the start bit is a 1. Since bit 0 (rightmost bit) is sent first, the ASCII Carriage Return character (CR) is sufficient to enable proper speed sense.

The following code snippet returns the values for DLH and DLL in r9, r8:

```
       MOV     r0, #1  ; dummy write data
       STRB    r0, [r3, #U_ASR]; Start autobaud sense (r3 as uart_base)
; now poll the autobaud sense register MSB until Valid is true
wait4valid
       LDR     r2, [r3, #U_ASR]; Read ASR
       TST     r2, #0x80000000 ; the Valid bit (active high)
       BEQ     wait4valid
; autobaud sense check complete...
; r2 as number of 1x clocks in one bit time
; representing the number of 24MHz clocks in the start bit
; Since this represents 16 of the baud (16x) clocks, we
; will be dividing by 16 to get baud divisor, but first
; round to nearest by adding 8 before the divide:
       ADD     r4, r2, #8 ; add 1/2 resolution
       MOV     r6, r4, LSR #4 ; divide by 16... R6 will have total divisor
       AND     r8, r6, #0xFF  ; copy DLL to r8
       MOV     r9, r6, LSR #8 ; copy DLM to r9
       AND     r9, r9, #0xff ; mask upper bytes
```

36.2.16 Baud Rate Generator

The following table given below is a divisor table for the baud rate, assuming the oscillator is 24.000 MHz.

Table 118: Baud Rate Generator Programming

Baud Rate	Divisor
300	5000
1200	1250
2400	625
4800	312
9600	156
19.2K	78
38.4K	39
57.6K	26
115.2K	13

36.2.17 Power-up Defaults

The following defines the Power-Up defaults:

- IER = 0
- ISR = 1
- LCR = 0
- MCR = 0
- LSR = 60 HEX
- MSR = Bits 0-3 = 0, Bits 4-7 = inputs
- FCR = 0
- TX = High

- OP1 = High
- OP2 = High
- RTS = High
- DTR = High
- RXRDY = High
- TXRDY = Low
- INT = Low

36.3 VFIR Programming Guidelines

36.3.1 IRDA (FIR/VFIR) and UART B

The IRDA module includes a fully functional 16550 compatible UART for implementing the SIR protocol. The UART pins will be selected automatically when the VFIR.CTL Mode bits are set to UART or SIR mode. Set the UART IRDA.CSR register to SIR mode to convert the UART signals to SIR signals.

If the IRDA.CTL Mode bits are set the FIR or VFIR modes, the UART pins will be disabled, and the FIR/VFIR pins will be used.

36.3.2 APB Bus Interface to Control/Status/FIFO

The Control and Status registers, and the FIFO reside on the APB bus. The processor programs the Control registers to begin transactions, and can read progress and error status from the Status Register. FIFO access is normally performed via the APB_DMA module.

36.3.3 FIR Mode Programming

1. Set the CLK_SOURCE_VFIR for 48 MHz and enable the UART_B Clock
2. Program IR.TX and IR.RX pin-mux settings to select IR (UART B)
3. Set the VFIR.CTL Mode to binary 100 (FIR)
4. FIR mode is half-duplex. Set the VFIR.CTL Transmit bit to '1' for transmits, '0' for receives
5. Read the VFIR.STS register to clear inputs
6. Program the APB_DMA to transfer data to/from the VFIR module
7. Enable the DMA bit in the VFIR.CTL register
8. Enable interrupts as needed
9. Set the GO bit in the VFIR.CTL register to start the action
10. Wait for DONE status or an interrupt which indicates that the transaction has completed
11. Check for errors when reading the VFIR Status and Interrupt Identification register
12. Clear the status bits by writing a '1' to them before starting the next packet.
13. For receive operations, the Incoming Frame Data Length (IFDL) register will contain the number of bytes received. DMA should have read this many bytes from the FIFO.

36.3.4 VFIR Mode Programming

1. Set the CLK_SOURCE_VFIR for 48MHz and enable the UART_B Clock
2. Program IR.TX and IR.RX pin mux settings to select IR (UART B)
3. Set the VFIR.CTL Mode to binary 101 (VFIR)
4. VFIR mode is half-duplex. Set the VFIR Control register Transmit bit to '1' for transmits, and '0' for receives

5. Read the VFIR.STS register to clear inputs
6. Program the APB_DMA to transfer data to/from the VFIR module
7. Enable the DMA bit in the VFIR.CTL register
8. Enable interrupts as needed
9. Set the GO bit in the VFIR.CTL register to start the action
10. Wait for DONE status or an interrupt which indicates that the transaction has completed
11. Check for errors when reading the VFIR Status and Interrupt Identification register
12. Clear the status bits by writing a '1' to them before starting the next packet.
13. For receive operations, the Incoming Frame Data Length (IFDL) register will contain the number of bytes received. DMA should have read this many bytes from the FIFO.

36.3.5 COM-SLAVE Interface (UART B)

The COM-SLAVE interface consists of a UART, which provides serial data synchronization and parallel-to-serial and serial-to-parallel data conversion for both receiver and transmitter sections. Synchronization for the serial data stream is accomplished by adding start and stop bits to the transmit data to form a data character. Data integrity is accomplished by attaching a parity bit to the data character. The parity bit can be checked by the receiver for any transmission bit errors.

The COM-SLAVE interface is fully programmable by an 8-bit CPU interface. The registers are 32-bit Word aligned. The interface supports word lengths from 5 to 8 bits, an optional parity bit, and 1 or 2 stop bits. If enabled, the parity can be odd, even, or forced to a defined state. Interrupts can be generated from any of 10 sources.

The UART supports both 16450 and 16550 compatible modes. The default mode is the 16450. The 16550 mode, which provides independent 16-byte FIFOs for transmits and receives, is selected via the FIFO control register. It also includes a 16-bit programmable baud rate generator and an 8-bit Scratch register, 8 modem control lines, and two DMA handshake lines that are used to indicate when the FIFOs are ready to transfer data to the CPU.

Table 119: Serial Bus Interface Signals for UART B

Signal Name	Direction	Description
UB3_RXD	I	Receiver Data Port
UB3_TXD	O	Transmit Data Port

36.3.6 SIR Pulse Encoder/Decoder

The UART transmit (TX) data is passed through this module prior to being muxed out to the IR transmitter. This module converts transmitted zeros into a 3/16 Return-To-Zero (RZ) pulse.

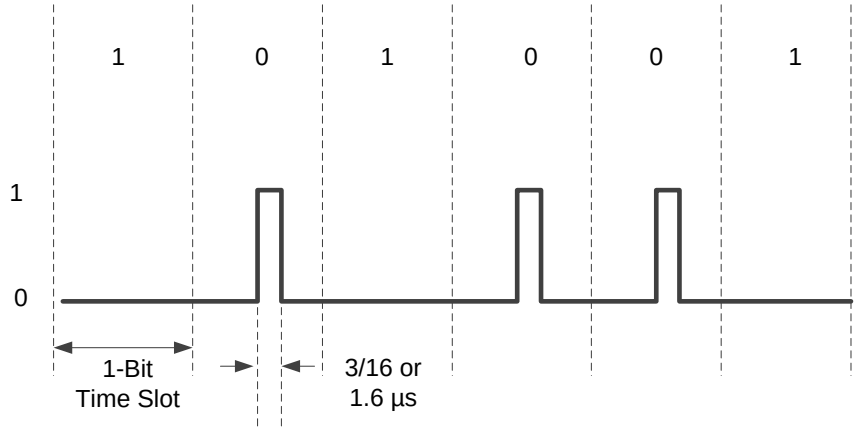
Similarly, received (RX) data is bit-synchronized using the 16X baud clock and the original serial data recovered from the IR bit stream. This data is then sent to the UART for reception.

The signal generated in SIR is as follows:

- On logic '1', the LED is off.
- On logic '0', a pulse is created starting the center of the bit time and lasting 3/16 of a bit time period or 1.6µs (3/16 bit times at 115.2 kbps) depending on the current settings.

The figure below displays the output for the input 101001 (in sending order) in SIR encoding.

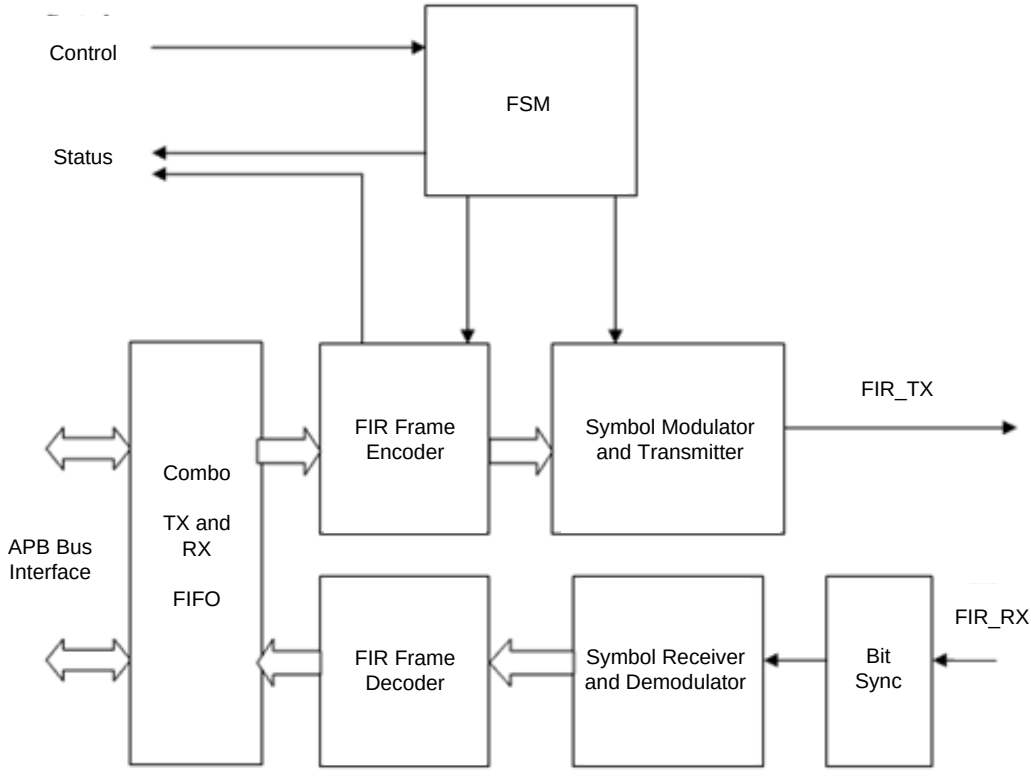
Figure 146: Output in Sending Order in SIR Encoding



36.3.7 FIR Encoder/Decoder

The figure below shows the micro-architecture for FIR and VFIR modes. Although the architecture is the same, the FIR and VFIR encoding schemes are very different.

Figure 147: Architecture for FIR and VFIR Modes



The Frame Level Encoder calculates CRC for the sent data, performs bit stuffing, creates start and stop sequences, and sends data down the line to the Data Shift Register. The Frame Level Decoder calculates the CRC on incoming data, performs bit de-stuffing, compares the calculated CRC to the received one, and reports any errors detected in the process.

The 1-symbol modulator creates the signal to be sent to the IR transceiver bit by bit as received from the UART in SIR mode, 2-bit signals in FIR mode, and 3-bit signals in VFIR mode.

FIR mode uses a Pulse Position Modulation code called 4PPM. The encoding sends one symbol every four time slices called "chips". Because there are four unique chip positions within each symbol in 4PPM, four independent symbols exist in which only one chip is logically a "one" while all other chips are logically a "zero." These four unique symbols are defined to be the only legal data symbols (DD) allowed in 4PPM. Each DD represents two bits of payload data, so that a byte of payload data can be represented by four DDs in sequence. The table below defines the chip pattern representation of the four unique DDs defined for 4PPM.

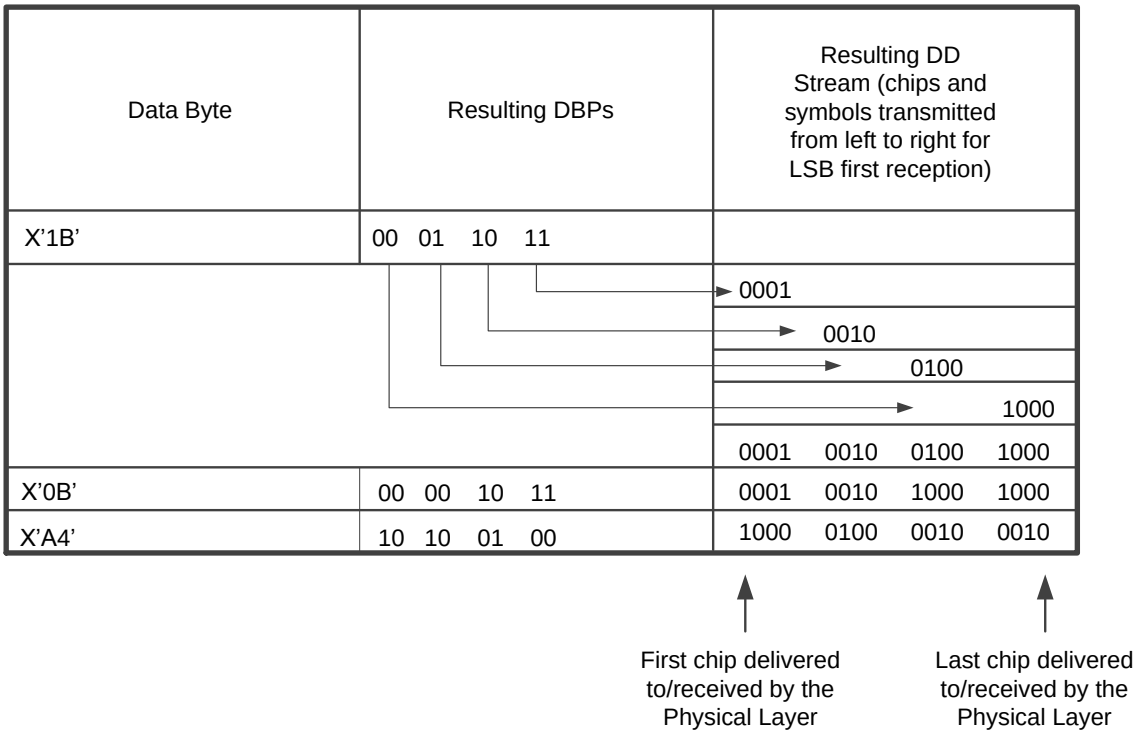
Table 120: Chip Pattern

Data Bit Pair	4PPM Data Symbol
00	1000
01	0100
10	0010
11	0001

A logical "1" represents a chip duration when the transmitting LED is emitting light, while a logical "0" represents a chip duration when the LED is off. Data encoding for transmission is done LSB first.

The figure below shows how various data bytes would be represented after encoding for transmission. In these examples transmission time increases from left to right so that chips and symbols farthest to the left are transmitted first.

Figure 148: Various Data Bytes After Encoding for Transmission



36.3.8 PPM Packet Format

For FIR 4PPM packets the data rate is 4 Megabits per second, and the signaling rate is 8.0 Mchips per second, where a chip is the smallest element of IrDA signaling. The packet format is defined below.

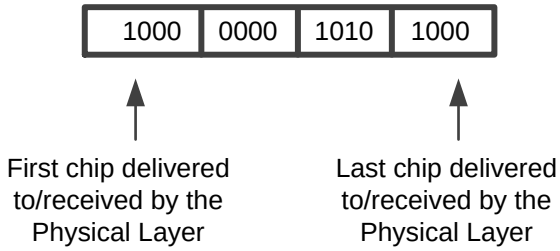
In this packet format, the payload data is encoded as described in the 4PPM encoding above, and the encoded symbols reside in the DD field. Maximum packet length is negotiated by the same mechanism as for the slower rates. The preamble field (PA) is used by the receiver to establish phase lock. During PA, the receiver begins to search for the start flag (STA) to establish symbol synchronization. If STA is received correctly, the receiver can begin to interpret the data symbols in the DD field. The receiver continues to receive and interpret data until the stop flag (STO) is recognized. STO indicates the end of a frame. The chip patterns and symbols for PA, STA, FCS field, and STO are defined below. Only complete packets that contain the entire format defined above are guaranteed to be decoded at the receiver (as for the lower rates, the information field, I, may be of zero length).

The 4PPM data encoding described above defines only the legal encoded payload data symbols. All other chip combinations are by definition illegal symbols for encoded payload data. Some of these illegal symbols are used in the definition of the preamble, start flag, and stop flag fields because they are unambiguously not data.

36.3.8.1 Preamble Field Definition

The preamble field (PA) consists of exactly 16 repeated transmissions of the following stream of symbols. In the PA field, transmission time increases from left to right so that chips and symbols on the left are transmitted first. See figure below.

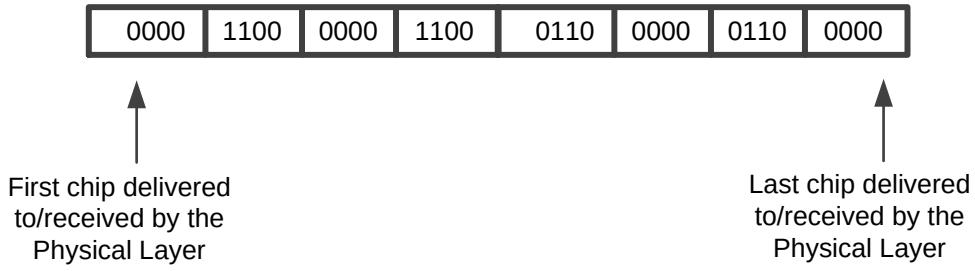
Figure 149: Preamble Field



36.3.8.2 Start Flag Definition

The start flag (STA) consists of exactly one transmission of the following stream of symbols. In the STA field, transmission time increases from left to right so that chips and symbols on the left are transmitted first. See figure below.

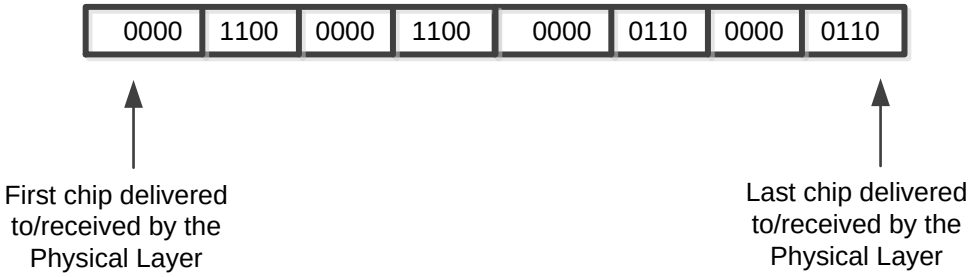
Figure 150: Start Flag



36.3.8.3 Stop Flag Definition

The stop flag (STO) consists of exactly one transmission of the following stream of symbols. In the STO field, transmission time increases from left to right so that chips and symbols on the left are transmitted first. See figure below.

Figure 151: Stop Flag



36.3.8.4 Frame Check Sequence Field Definition

The frame check sequence (FCS) field is a 32-bit field that contains a cyclic redundancy check (CRC) value. The CRC is a calculated, payload data dependent field, calculated before 4PPM encoding. It consists of the 4PPM encoded data resulting from the IEEE 802 CRC32 algorithm for cyclic redundancy check as applied to the payload data contained in the packet. The CRC32 polynomial is defined as follows:

$$CRC(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

The CRC32 calculated result for each packet is treated as four data bytes, and each byte is encoded in the same fashion as is payload data. Payload data bytes are input to this calculation in LSB first format. The 32-bit CRC register is preset to all "1's" prior to calculation of the CRC on the transmit data stream. When data has ended and the CRC is being shifted for transmission at the end of the packet, a "0" should be shifted in so that the CRC register becomes a virtual shift register.

Note:	The inverse of the CRC register is what is shifted as defined in the polynomial.

36.3.8.5 Aborted Packets

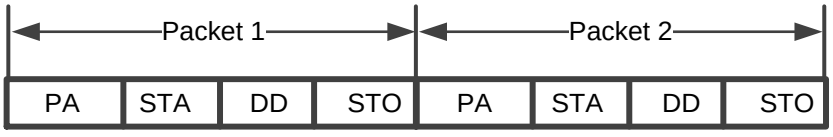
Receivers may only accept packets that have valid STA, DD, FCS, and STO fields as defined in the PPM Packet Format section. The PA field need not be valid in the received packet. All other packets are aborted and ignored.

Any packet may be aborted at any time after a valid STA but before transmission of a complete STO flag by two or more repeated transmissions of the illegal symbol "0000." Also, any packet may be aborted at any time after a valid STA by reception of any illegal symbol which is not part of a valid STO field.

36.3.8.6 Back to Back Packet Transmission

Back to back, or "brick-walled" packets are allowed, but each packet must be complete (i.e., containing PA, STA, DD, and STO fields). Brick-walled packets are illustrated in the figure below.

Figure 152: Back-to-Back (Brick-Walled) Packet Transmission



36.3.9 VFIR 16.0 Megabits per Second Rate

The 16.0 Megabits per second data transmission incorporates a new modulation code HHH (1,13) - low duty cycle, rate 2/3, (d,k) = (1, 13) run-length limited (RLL) code to achieve the specified data rate from a 24MHz reference clock. The HHH(1,13) code guarantees for at least one empty chip and at most 13 empty chips between chips containing pulses in the transmitted IR signal. The data transmission packet/frame is based on the 4.0 Megabits per second frame format with modifications introduced where necessary to accommodate the requirements that are specific to the new modulation code. The system includes a simple scrambling/descrambling scheme to randomize the duty cycle statistics. The signaling rate of the 16.0 Megabits per second data rate is 24.0 Mchips per second, where a chip is the smallest element of IrDA signaling.

36.3.10 HHH (1, 13) Modulation Code

The HHH (1, 13) modulation code has the following salient features:

- Code Rate: 2/3 ,
- Maximal Duty Cycle: 1/3 (~33%) ,
- Average Duty Cycle: ~26% ,
- Minimal Duty Cycle: 1/12 (~8.3%) ,
- Run-Length Constraints: (d, k) = (1, 13) ,
- Longest Run of '10's: yyy'000'101'010'101'000'yyy ,
- Chip Rate @ Data Rate 16 Megabits per second: 24 Mchips per second ,
- System Clock @ Data Rate 16 Megabits per second: N X 12 MHz (where N ≥ 4)

The HHH(1,13) code is a Run Length Limited (RLL) code that provides both power efficiency and bandwidth efficiency at the high data rate. The signaling rate of the code is 24 Mchips per second allowing a rise and fall time of 19 ns. LED on time is further improved by having a 26% average duty cycle for random data. The lower duty cycle is achieved by scrambling the incoming data stream. The run length constraints (d, k) = (1, 13) ensure an inactive chip after each active chip, i.e. only single-chip-width pulses occur. This feature allows a source or a receiver to exhibit a long tail property. To take full advantage of the d = 1 feature of HHH(1, 13) in strong signal conditions, clock and data recovery circuitry ignores the level of the chip following an active chip and assumes these chips are inactive. The modulation code is enhanced with simple frame-synchronized scrambler/descrambler mechanisms as defined and described in later in this chapter.

36.3.10.1 HHH Data Encoding Definition

The encoding definition of the HHH (1, 13) code is provided by a state transition table described as follows:

Specific data pair D ≡ □D* = (d1, d2) arriving at the encoder input is first associated with a corresponding next state N ≡□N*. This occurs as soon as the data D* bits have advanced into the positions of the internal data bits B1 = (b1, b2), i.e., when (b1, b2, b3, b4, b5, b6) is identical to (d1, d2, x, x, x, x). In a second step, during the next encoding cycle, the state S takes on the value of N*, i.e., S ≡ S* ← □N* so that S is now associated with (d1, d2). In the same cycle, the inner codeword C ≡ □C* now carrying the information of D* is computed. Thus, referring the figure below, a given internal input vector (b1, b2, b3, b4, b5, b6) associates the bits (b1, b2) with the next state N and a given state S associates the data pair ahead of (b1, b2) to the output C. In other words, the pair-wise values for N and C as listed in the figure below are not associated with the same input data pair.

Encoder initialization: The state S = (s1, s2, s3) = (1, 0, 0) is also used as the initial state of the encoder, i.e., denoting with (α, β), the first pair of data bits to be encoded, the state S is forced to take on the value (1, 0, 0) when the bits (α, β) have advanced into the encoding circuits such that the internal inputs B1 = (b1, b2) is identical to (α, β).

Table 121: Encoder Initialization

Present State: S = (s_1, s_2, s_3)	Next State/Internal Output: N = (n_1 n_2, n_3) / C = (c_1, c_2, c_3)							
	Internal Inputs: (b_1, b_2, b_3, b_4, b_5, b_6)							
	00xxxx	01xxxx	10xxxx	1100xx	1101xx	111011	1110(~~11~~)	1111xx
000	000/010	001/010	010/010	111/010	111/001	111/010	011/010	011/010
001	000/001	001/001	100/001	100/010	100/010	100/010	100/010	100/010
010	000/100	001/100	010/100	111/100	111/101	111/100	011/100	011/100
011	000/101	001/101	100/101	100/100	100/100	100/100	100/100	100/100
100	000/000	001/000	010/000	011/000	011/000	011/000	011/000	011/000
111	100/000	100/000	111/000	100/000	100/000	100/000	100/000	100/000

Note: Refer to the IrDA specification 1.4 for further details.

36.3.10.2 HHH (1, 13) Packet Format

The packet format for 16.0 Megabits per second HHH(1,13) has the following form.

Figure 153: HHH (1, 13) Packet Format

The payload data is encoded as described in the HHH (1,13) encoding above, and the encoded symbols reside in the IrLAP Frame field. The preamble field (PA) is used by the receiver to establish phase lock. During PA, the receiver begins to search for the start flag (STA) to establish symbol synchronization. If STA is received correctly, the receiver can begin to interpret the data symbols in the IrLAP Frame field. The receiver continues to receive and interpret data until the stop flag (STO) is recognized. STO indicates the end of a frame. The chip patterns and symbols for PA, STA, CRC field, and STO are defined below.

Only complete packets that contain the entire format defined above are guaranteed to be decoded at the receiver (as for the lower rates, the information field, I, that is part of the IrLAP field, may be of zero length).

The 16.0 Megabits per second packet contains several fields for the purposes of clock recovery, synchronization, and data transmission. In concept, the packet format is similar to that used in 4.0 Megabits per second; however, there are specific controller elements like clock recovery, synchronization, and encoding/decoding circuits that need to be implemented specifically for 16.0 Megabits per second data rate.

36.3.10.3 Preamble Field Definition

The transmitted PREAMBLE (PA) is constructed by concatenating ten times (10X) the 24-chip (1 µs) PREAMBLE PERIOD (PP), where

PP = '100'010'010'001'001'001'000'100'

to form the complete 240-chip (10 µs) preamble

PA = 'PP'PP'PP'PP'PP'PP'PP'PP'PP'PP'

The left-most/right-most chip of PP and PA, respectively, is transmitted first/last and a '1' in PP means an active chip (pulse) and a '0' means an empty chip (no pulse).

36.3.10.4 Start (STA) Flag Definition

The transmitted START (STA) delimiter is the 48-chip (2 µs) chip sequence

STA = '100'101'010'100'100'010'000'001'001'010'101'001'000'001'010'000'

The left-most/right-most chip of STA is transmitted first/last and a '1' in STA means an active chip (pulse) and a '0' means an empty chip (no pulse). The Start Flag Delimiter allows for packet synchronization. A delimiter detection circuit should declare a flag as having been found when there is a perfect match between the receiver chip stream and a particular delimiter. The Start and Stop delimiters contain a subsequence '1001010101001' that violates the HHH (1,13) code. This subsequence occurs twice in the Start Flag delimiter and never occurs within the main HHH code.

36.3.10.5 IrLAP Frame

The structure remains unchanged from that defined in the IrLAP Specification, Version 1.1. The content of the IrLAP frame is first scrambled and then encoded with HHH(1,13). Note that the 32 CRC bits for the IrLAP frame are calculated before the IrLAP frame is scrambled. For reference, the IrLAP frame as the following structure:

| Address (8 bits) | Control (8 bits) | Information (M times 8 bits) |

36.3.10.6 CRC

Computation remains unchanged from the 32-bit CRC defined for the 4 Megabits per second data rate. The content of the CRC field is first scrambled with the scheme described later and then encoded with HHH(1, 13) as described previously. Note that the 32 CRC bits for the IrLAP frame are calculated before the IrLAP frame is scrambled. The transmitted CRC field is a 48-chip (2 µs) sequence.

36.3.10.7 FLUSH BYTE (FB)

The Flush Byte (FB) is the 8-bit sequence:

FB = '00'00'00'00'.

These 8 bits are not scrambled but directly sent to the HHH(1,13) encoder. The transmitted FB field is a 12-chip (0.5 µ s) sequence. Note that the FB field is required to enable complete decoding of the CRC field. The flush byte denotes the end of the main body. Since the flush byte is not scrambled, a well-balanced HHH(1,13) sequence precedes the STOP delimiter. This sequence would be equivalent to the UART "Break" command.

36.3.10.8 STOP (STO)

The transmitted STOP (STO) delimiter is the 48-chip (2 µs) sequence:

STO = '001'001'010'101'001'000'100'000'100'101'010'100'100'000'100'000'.

The left-most/right-most chip of STO is transmitted first/last and a '1' in STO means an active chip (pulse) and a '0' means an empty chip (no pulse). As in the Start Flag delimiter, the Stop flag also contains a subsequence '1001010101001' that violates the HHH (1,13) code. This subsequence also occurs twice in the Stop Flag delimiter.

36.3.10.9 NULL Sequence

The transmitted NULL sequence is the 24-chip (1 µs) sequence

NULL = '000'000'000'000'000'000'000'000'.

The NULL field is a new field for the purpose of providing an HHH(1,13) code pattern violation that permits terminating reception of the packet in the event that the STO field is not recognized. The left-most/right-most chip in NULL is transmitted first/last and all chips of NULL are empty chips (no pulses). The NULL field increases the probability that the packet is terminated close to the STOP flag. The NULL field also reduces the probability that two back to back packets are interpreted as a single packet, should the STOP flag delimiter of the first packet be missed.

36.3.10.10 Scrambling and Descrambling Functions

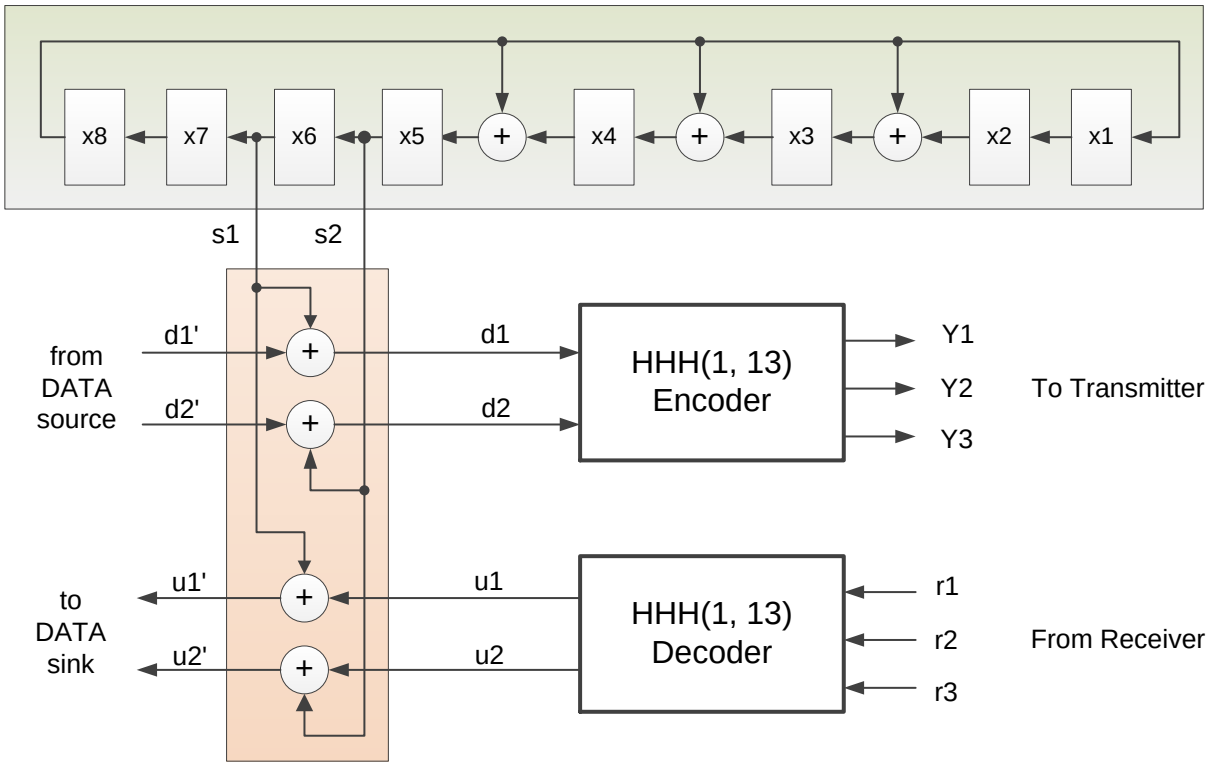
It is advantageous to enhance the encoder/decoder system with simple scrambler/descrambler functions.

The primitive polynomial: $x^8 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$,

where $\oplus$ indicates a modulo-2 addition or, equivalently, a logic exclusive OR (XOR) operation. The operations of the proposed scrambling and descrambling functions are performed according to the principles of frame synchronized scrambling/descrambling (FSS) mechanisms. The scrambling/descrambling scheme is shown in the figure below. The linear feedback shift register (LFSR) produces a maximum-length pseudo-random sequence with period 255. The output of register cell x6 shown in the figure is defined to be the equivalent serial output of the LFSR.

The modulo-2 adders shown in the figure below correspond to logic XOR (exclusive OR) gates. During transmission, each new pair of source bits (d1', d2') is XOR-ed with a new pair of scrambling bits (s1, s2) to produce the scrambled data bit pair (d1, d2) entering the encoder. Similarly, during reception, each new pair of decoded bits (u1, u2) is XOR-ed with a new pair of descrambling bits (s1, s2) to produce the descrambled user bit pair (u1', u2') that is sent to the data sink. A scrambling/descrambling cycle has duration 3T seconds where T = 41.7 ns is the chip period.

Figure 154: Scrambling / Descrambling Scheme



36.3.10.11 Effects and Limits of Scrambling/Descrambling

By enhancing the system with scrambling/descrambling functions during data transmission/reception, one achieves generally better duty cycle statistics in the HHH(1,13) coded channel chip stream; the resulting duty cycle converges towards the average duty cycle of the code (≈ 26%) for typical payload data. Scrambling can greatly reduce the probability of occurrence of worst-case patterns.

36.3.10.12 Aborted Packets

Receivers may only accept packets that have valid STA, IrLAP frame, CRC, and STO fields as defined in the Packet Format section. The PA field need not be valid in the received packet. All other packets are aborted and ignored.

36.3.10.13 Back to Back Packet Transmission

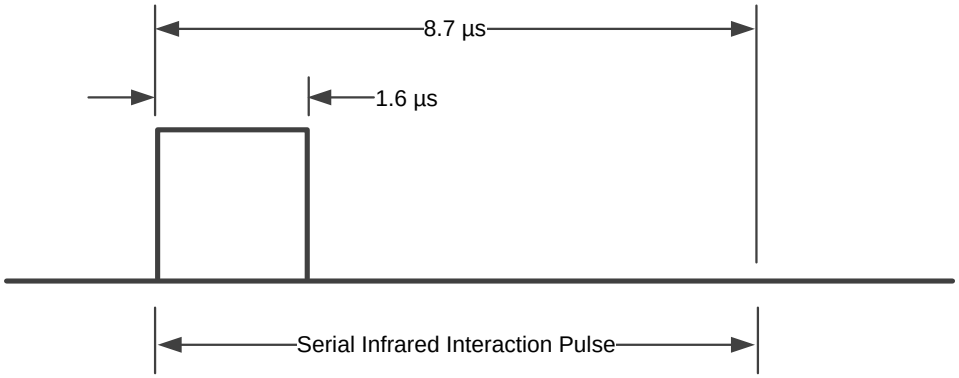
Back to back, or "brick-walled" packets are allowed, but each packet must be complete (i.e., containing PA, STA, IrLAP Frame, CRC and STO fields).

36.3.11 SIR - Serial Interaction Pulse

In order to guarantee non-disruptive coexistence with slower (up to 115.2 Kilobits per second) systems, once a higher speed (above 115.2 Kilobits per second) connection has been established, the higher speed system must emit a Serial infrared Interaction Pulse (SIP) at least once every 500 ms as long as the connection lasts to quiet slower systems that might interfere with the link. The pulse can be transmitted immediately after a packet has been transmitted. Initiation of this pulse will be performed by software by setting a bit in the VFIR Control register.

The pulse is shown below.

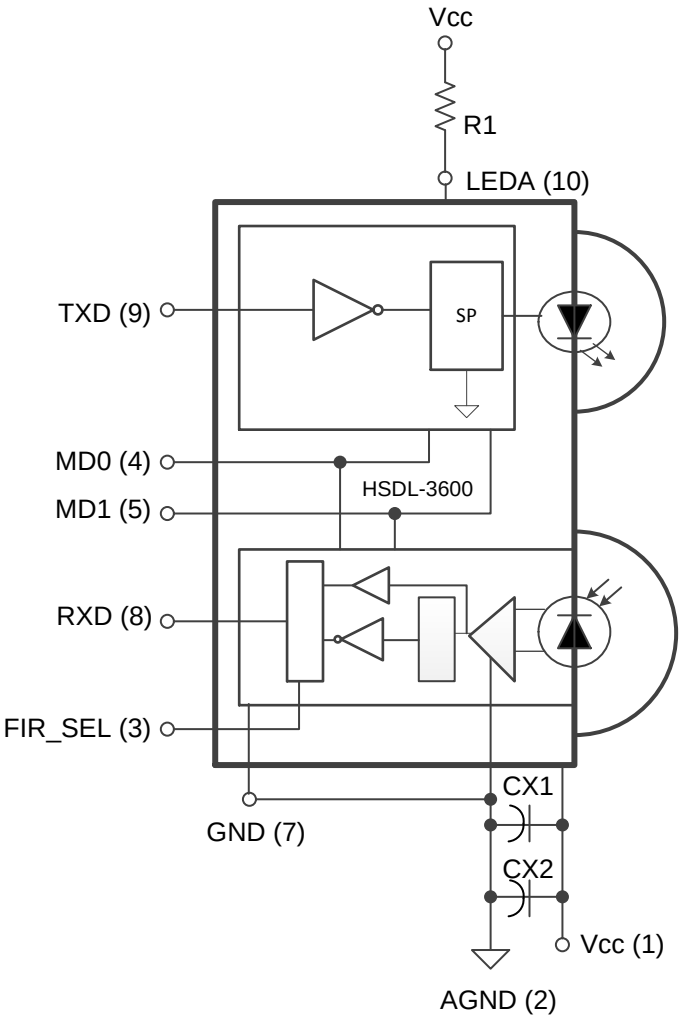
Figure 155: SIP



36.3.11.1 External Interface

The SIR/FIR/VFIR signals are muxed to the external transceiver. The transceiver has TX and RX pins, and may optionally have controls for receiver gain and transmitter power. This module provides the TX and RX pins, and each has a polarity control for compatibility with all vendors. Other transceiver pins are not directly supported, although they could be provided using GPIO signals. Refer to the drawing below for typical interface. TXD and RXD are VFIR module pins. MD and FIR_SEL pins would be GPIO. MD pins are for transmit power. FIR_SEL controls gain of receiver, and should be set to zero for SIR mode.

Figure 156: Typical Interface



36.4 UART Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

The Tegra 4 UART is based on the 16550 industry standard Universal Asynchronous Receiver/Transmitter (UART), with enhancements to support autobaud detection and End-of-Received Data timeout detection.

The UART supports a device clock of up to 72 MHz, for a maximum baud rate of 4.5 Megabits per second.

The THR, RBR and DLL registers all occupy the same address space.

- The Transmitter Holding Register (THR) is Write-Only, and can be accessed if the LSR.DLAB bit is clear.
- The Receiver Buffer Register (RBR) is Read-Only, and can be accessed if the LSR.DLAB bit is clear.
- The Divisor Latch LSByte Register (DLL) is Read/Write and can be accessed if the LSR.DLAB bit is set.

36.4.1 UART_THR_DLAB_0_0

UART Transmit Holding Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	R/W	Reset	Description
7:0	RO	0x0	THR_A: Transmit holding register, holds the character to be transmitted by the UART. In FIFO mode, a write to this FIFO places the data at the end of the FIFO.
7:0	RO	X	RBR_A: Receive Buffer Register. Rx Data read from here.
7:0	RO	0x0	DLL_A: Divisor Latch LSB (low 8 bits of 16-bit Baud Divisor)

36.4.2 UART_IER_DLAB_0_0

The IER and DLH registers occupy the same address space, selected by the LSR.DLAB bit. The Interrupt Enable Register (IER) is selected if the LSR.DLAB bit is clear. The Divisor Latch MSByte register (DLM) is selected if the LSR.DLAB bit is set. The DLM register holds the upper 8 bits of the 16-bit Baud Divisor (16x).

UART Interrupt Enable por=0x00000000

RX_TIMEOUT occurs when data has been sitting in the Rx FIFO for more than 4 character times without being read because there is not enough data to reach the trigger level. Interrupt needed to handle this case so that all data is received in a timely manner. Note that this normally occurs at the end of an incoming data stream.

EORD (End of Receive Data) Interrupt occurs when the receiver detects that data stops coming in for more than 4 character times. This interrupt is useful for determining that the sending device has completed sending all its data. EORD timeout will not occur if the receiving data stream is stopped because of hardware handshaking.

To clear the EORD timeout interrupt you must DISABLE the EORD interrupt enable (IE_EORD).

Interrupt Enable and Divisor Latch MSByte Registers

Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx000000)

Bit	Reset	Description
5	0x0	IE_EORD: Interrupt Enable for End of Received Data 0 = DISABLE 1 = ENABLE
4	0x0	IE_RX_TIMEOUT: Interrupt Enable for Rx FIFO timeout 0 = DISABLE 1 = ENABLE
3	0x0	IE_MSI: Interrupt Enable for Modem Status Interrupt 0 = DISABLE 1 = ENABLE
2	0x0	IE_RXS: Interrupt Enable for Receiver Line Status Interrupt 0 = DISABLE 1 = ENABLE
1	0x0	IE_THR: Interrupt Enable for Transmitter Holding Register Empty interrupt 0 = DISABLE 1 = ENABLE
0	0x0	IE_RHR: Interrupt Enable for Received Data Interrupt 0 = DISABLE 1 = ENABLE

36.4.3 UART_IIR_FCR_0

The FCR and IIR registers occupy the same address space. The FIFO Control Register (FCR) is Write-Only. The Interrupt Identification Register (IIR) is Read-Only

UART FIFO Control Register por=0x00000000

The DMA can run in one of two modes:

- If Mode0 is selected (NO_CHANGE) the Rx DMA request will go active whenever the Rx FIFO is not empty. Only single byte transactions can be performed in this mode. If the FIFO is not enabled, Mode0 MUST be used.
- If Mode1 is selected (CHANGE) the Rx DMA request will go active whenever the Rx FIFO trigger level is reached.

For best performance, always enable the FIFO and select DMA Mode 1.

For RX_TRIG the FIFO_COUNT references the number of bytes in the receive FIFO.

For TX_TRIG the FIFO_COUNT references the number of empty bytes in the transmit FIFO. For example, FIFO_COUNT_GREATER_16 means that there are at least 16 empty byte slots in the TX_FIFO

UART Interrupt ID Register por=0x00000001

The Interrupt ID field indicates the current highest priority interrupt. If more than one interrupt is pending the one with the highest priority will be shown.

The table below shows the encoding. Priority flows from top (highest) to bottom (lowest).

Table 122 Interrupt ID Encoding

IIR[3:0]	Priority Level
0001	No interrupt
0110	Overrun Error, Parity Error, Framing Error, Break
0100	Receiver Data Available
1100	rx_timeout_intr
1000	eord_timeout_intr
0010	Transmitter Holding Register empty
0000	modem_status interrupt

UART FIFO Control and Interrupt Identification Registers

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:6	X	EN_FIFO: FIFO Mode Status 0=16450 mode(no FIFO) 1 = 16550 mode (FIFO) 1 = MODE_16550 0 = MODE_16450
7:6	0x0	RX_TRIG: 0 = FIFO_COUNT_GREATER_1 1 = FIFO_COUNT_GREATER_4 2 = FIFO_COUNT_GREATER_8 3 = FIFO_COUNT_GREATER_16

Bit	Reset	Description
5:4	0x0	TX_TRIG: 0 = FIFO_COUNT_GREATER_16 1 = FIFO_COUNT_GREATER_8 2 = FIFO_COUNT_GREATER_4 3 = FIFO_COUNT_GREATER_1
3	X	IS_PRI2: Encoded Interrupt ID Refer to IIR[3:0] table above 0 = DISABLE 1 = ENABLE
3	0x0	DMA: 0:DMA_MODE_0 1:DMA_MODE_1 0 = NO_CHANGE 1 = CHANGE
2	X	IS_PRI1: Encoded Interrupt ID Refer to IIR[3:0] table above 0 = DISABLE 1 = ENABLE
2	0x0	TX_CLR: 1 = Clears the contents of the transmit FIFO and resets its counter logic to 0 (the transmit shift register is not cleared or altered). This bit returns to 0 after clearing the FIFOs. 0 = NO_CLEAR 1 = CLEAR
1	X	IS_PRI0: Encoded Interrupt ID Refer to IIR[3:0] table above 0 = DISABLE 1 = ENABLE
1	0x0	RX_CLR: 1 = Clears the contents of the receive FIFO and resets its counter logic to 0 (the receive shift register is not cleared or altered). This bit returns to 0 after clearing the FIFOs. 0 = NO_CLEAR 1 = CLEAR
0	X	IS_STA: Interrupt Pending if ZERO 0 = INTR_PEND 1 = NO_INTR_PEND
0	0x0	FCR_EN_FIFO: 1 = Enable the transmit and receive FIFOs. This bit should be enabled 0 = DISABLE 1 = ENABLE

36.4.4 UART_LCR_0

UART Line Control Register

Offset: 0xc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	DLAB: Divisor Latch Access Bit (set to allow programming of the DLH, DLM Divisors) 0 = DISABLE 1 = ENABLE
6	0x0	SET_B: Set BREAK condition -- Transmitter will send all zeroes to indicate BREAK 0 = NO_BREAK 1 = BREAK

Bit	Reset	Description
5	0x0	SET_P: Set (force) parity to value in LCR [4] 0 = NO_PARITY 1 = PARITY
4	0x0	EVEN: Even parity format. There will always be an even number of 1s in the binary representation (PAR = 1) 0 = DISABLE 1 = ENABLE
3	0x0	PAR: 0 = No parity sent 0 = NO_PARITY 1 = PARITY
2	0x0	STOP: 0 = Transmit 1 stop bit 1 = Transmit 2 stop bits (receiver always checks for 1 stop bit) 0 = DISABLE 1 = ENABLE
1:0	0x0	WD_SIZE: 3=Word length of 8 0 = WORD_LENGTH_5 1 = WORD_LENGTH_6 2 = WORD_LENGTH_7 3 = WORD_LENGTH_8

36.4.5 UART_MCR_0

The RTS and CTS pins are used for hardware handshaking with and external serial device. RTS (Request-To-Send) informs the device that the UART is ready to accept data. CTS (Clear-To-Send) comes from the RTS of the external device and indicates that data can be sent.

The RTS pin can be controlled manually with the RTS bit in the MCR, or automatically by setting the RTS_EN bit.

If RTS_EN is set, the RTS pin will automatically remove the Request-To-Send when the FIFO reaches the trigger level.

If the CTS_EN bit is set, the UART transmitter will stop transmitting when the Clear-To-Send input is taken away.

UART Modem Control Register

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000000)

Bit	Reset	Description
6	0x0	RTS_EN: 1 = Enable RTS Hardware Flow Control 0 = DISABLE 1 = ENABLE
5	0x0	CTS_EN: 1 = Enable CTS Hardware Flow Control 0 = DISABLE 1 = ENABLE
4	0x0	LOOPBK: 1 = Enable internal loop back of Serial Out to In 0 = DISABLE 1 = ENABLE
3	0x0	OUT2: nOUT2 (Not Used) 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	OUT1: nOUT1 (Not Used) 0 = DISABLE 1 = ENABLE
1	0x0	RTS: 0 = Force RTS to high if RTS hardware flow control not enabled 0 = FORCE_RTS_HI 1 = FORCE_RTS_LOW
0	0x0	DTR: 1 = Force DTR to high 0 = FORCE_DTR_HI 1 = FORCE_DTR_LOW

36.4.6 UART_LSR_0

UART Line Status Register

Offset: 0x14 │ Read/Write: RO │ Reset: 0x00000XXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
8	X	TX_FIFO_FULL: Transmitter FIFO full status 0 = NOT_FULL 1 = FULL
7	X	FIFOE: 1 = Receive FIFO Error 0 = NO_ERR 1 = ERR
6	X	TMTY: Transmit Shift Register empty status 0 = NO_EMPTY 1 = EMPTY
5	X	THRE: 1 = Transmit Holding Register is Empty -- OK to write data 0 = FULL 1 = EMPTY
4	X	BRK: 1 = BREAK condition detected on line 0 = NO_BREAK 1 = BREAK
3	X	FERR: 1 = Framing Error 0 = NO_FRAME_ERR 1 = FRAME_ERR
2	X	PERR: 1 = Parity Error 0 = NO_PARITY_ERR 1 = PARITY_ERR
1	X	OVRF: 1 = Receiver Overrun Error 0 = NO_OVERRUN_ERROR 1 = OVERRUN_ERROR
0	X	RDR: 1 = Receiver Data Ready (Data available to read) 0 = NO_DATA_IN_FIFO 1 = DATA_IN_FIFO

36.4.7 UART_MSR_0

UART Modem Status Register

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7	0x0	CD: State of Carrier detect pin 0 = DISABLE 1 = ENABLE
6	0x0	RI: State of Ring Indicator pin 0 = DISABLE 1 = ENABLE
5	0x0	DSR: State of Data set ready pin 0 = DISABLE 1 = ENABLE
4	0x0	CTS: State of Clear to send pin 0 = DISABLE 1 = ENABLE
3	0x0	DCD: Change (Delta) in CD state detected 0 = DISABLE 1 = ENABLE
2	0x0	DRI: Change (Delta) in RI state detected 0 = DISABLE 1 = ENABLE
1	0x0	DDSR: Change (Delta) in DSR state detected 0 = DISABLE 1 = ENABLE
0	0x0	DCTS: Change (Delta) in CTS state detected 0 = DISABLE 1 = ENABLE

36.4.8 UART_SPR_0

UART Scratch Pad Register

Offset: 0x1c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	SPR_A: Scratchpad register (not used internally)

36.4.9 UART_IRDA_CSR_0

Bits [7:6] control the infrared (SIR) encoding. If enabled, each zero bit is transmitted as a short IR pulse. No pulse is sent during idle or '1' data bits. The IR pulse can be either 3/16 or 4/16 of a normal bit time.

The low 4 bits control signal polarity and apply whether or not SIR coding is enabled.

UART IrDA Pulse Coding CSR Register

Offset: 0x20 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00xx0000)

Bit	Reset	Description
7	0x0	SIR_A: 1 = Enable SIR coder 0 = Disable SIR coder 0 = DISABLE 1 = ENABLE
6	0x0	PWT_A: 0=3/16th Baud Pulse, 1=4/16 0 = BAUD_PULSE_3_14 1 = BAUD_PULSE_4_14
3	0x0	INVERT_RTS: Inverts the normally inactive high nRTS pin 0 = DISABLE 1 = ENABLE
2	0x0	INVERT_CTS: Inverts the normally inactive high nCTS pin 0 = DISABLE 1 = ENABLE
1	0x0	INVERT_TXD: Inverts the normally inactive high TXD pin 0 = DISABLE 1 = ENABLE
0	0x0	INVERT_RXD: Inverts the normally inactive high RXD pin 0 = DISABLE 1 = ENABLE

36.4.10 UART_RX_FIFO_CFG_0

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx0x000001)

Bit	Reset	Description
7	0x0	EN_RX_FIFO_TRIG: Enable use of RX_FIFO_TRIG count (obsoletes RX_TRIG when enabled) 0 = DISABLE 1 = ENABLE
5:0	0x1	RX_FIFO_TRIG: Set RX_FIFO trigger level (any value 1 thru 32)

36.4.11 UART_MIE_0

UART Modem Interrupt Enable Register

Offset: 0x28 │ Read/Write: R/W │ Reset: 0x0000000f (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx1111)

Bit	Reset	Description
3	0x1	DCD_INT_EN: Interrupt Enable for Change (Delta) in CD state detected 0 = DISABLE 1 = ENABLE
2	0x1	DRI_INT_EN: Interrupt Enable for Change (Delta) in RI state detected 0 = DISABLE 1 = ENABLE
1	0x1	DDSR_INT_EN: Interrupt Enable for Change (Delta) in DSR state detected 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x1	DCTS_INT_EN: Interrupt Enable for Change (Delta) in CTS state detected 0 = DISABLE 1 = ENABLE

36.4.12 UART_ASR_0

The UART has autobaud sensing logic that can measure the width of the start bit of incoming data to determine baud rate. For this to work, the incoming character must have its LSB set. A <cr> works well (0x0d). The logic will use the UART device clock to count how wide the start pulse is and the BUSY bit will be set when the sensing is complete. The value in {RX_RATE_SENSE_H,RX_RATE_SENSE_L} should be divided by 16 with Round-to-Nearest, and the resulting value loaded into the DLM,DLL register pair to set the Baud Clock to 16X the Baud Rate.

UART Auto Sense Baud Register

Offset: 0x3c │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	VALID: This bit is set when the controller finishes counting the clocks between two successive clock edges after there is a write to ASR with don't care data. 0 = UN_SET 1 = SET
30	0x0	BUSY: This bit is set when there is a write to ASR and is reset when the controller finishes counting the clock edges between two successive clock edges. 0 = NO_BUSY 1 = BUSY
15:8	0x0	RX_RATE_SENSE_H: Shows bits [15:8] of the count of clock edges between two successive clock edges.
7:0	0x0	RX_RATE_SENSE_L: Shows bits [7:0] of the count of clock edges between two successive clock edges.

36.5 VFIR Registers

36.5.1 VFIR_CTL_0

The VFIR Control register is used to:

- enable/disable entire IRDA module
- configure the controller in FIR mode or VFIR mode or SIR or UART mode
- configure either in Transmit mode or Receive mode
- invert the polarity of Tx or Rx line
- invert the polarity of Tx or Rx line
- enable/disable DMA
- know the attention levels of Transmit FIFO and Receive FIFO
- start a transmit or receive operation
- select the frequency
- clear the Transmitter FIFO/Receiver FIFO

VFIR Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000020 (0b000xxxx0x000x000xx00000000100000)

Bit	Reset	Description
31	0x0	GLOBAL_ENABLE: 0 = Entire IRDA module is disabled 0 = DISABLE 1 = ENABLE
30	0x0	GO: Set to 1 to start transmit or receive operation, self-clearing 0 = STOP 1 = START
29	0x0	AUTO_RESTART: Set to 1 to restart another operation when previous is done 0 = STOP 1 = RE_START
24	0x0	FORCE_BAD_CRC: Debugging bit to force a CRC error 0 = DISABLE 1 = ENABLE
22	0x0	TX_FIFO_CLR: Clear the Transmitter FIFO 0 = DISABLE 1 = ENABLE
21:20	0x0	TX_ATN_LVL: 00: when not full 3 = slots_empty_12 2 = slots_empty_8 1 = slots_empty_4 0 = not_full
18	0x0	RX_FIFO_CLR: Clear the Receiver FIFO 0 = DISABLE 1 = ENABLE
17:16	0x0	RX_ATN_LVL: 00: when not empty 3 = slots_full_12 2 = slots_full_8 1 = slots_full_4 0 = not_empty
13	0x0	START_SIP: Self-clearing bit to initiate a Serial Ir Interaction Pulse.
12	0x0	EN_PULSECORR: Fixes single pulse errors caused by VFIR propagation effects. 0 = DISABLE 1 = ENABLE
11	0x0	DMA_EN: DMA Enable Allow requests to go to the APB_DMA controller. 0 = DISABLE 1 = ENABLE
10	0x0	TX_TERM: of outgoing frame, i.e., the calculated CRC will be sent, followed by the frame stop sequence. The interrupt will be generated if enabled. 0 = ABORT 1 = NORMAL
9	0x0	COUNT_ODATA: When enabled, the controller will end the frame when the Outgoing Frame Data Length (OFDL) count is reached. Use in conjunction with TX_Term above so controller will know when and how to end the frame. 0 = DISABLE 1 = ENABLE
8	0x0	FORCEBRK: break state (zero) regardless of what the transmitter is doing. 0 = TX_ENABLE 1 = TX_DISABLE

Bit	Reset	Description
7	0x0	NEGATE_RX: Invert polarity on incoming RX pin (applies to all modes) 0 = DISABLE 1 = ENABLE
6	0x0	NEGATE_TX: Invert polarity on outgoing TX pin (applies to all modes) 0 = DISABLE 1 = ENABLE
5:4	0x2	FREQUENCY: The 8 and 24 MHz clocks can be used to save power during transmits. The 72 MHz clock can be used in case the 48 MHz clock is not available due to system constraints. 0 = F8MHz 1 = F24MHz 2 = F48MHz 3= F72MHz
3	0x0	TRANSMIT: 0 = Receive (FIR/VFIR modes) 1 = TRANSMIT 0 = RECEIVE
2:0	0x0	MODE: 000: UART uses the UART B module 7 = RSVD 6 = RSVD1 5 = VFIR 4 = FIR 3 = RSVD2 2 = MIR 1 = SIR 0 = UARTB

36.5.2 VFIR_STS_0

VFIR Status and Interrupt Identification Register is used to know the status of:

- Transmitter/Receiver FIFO Trigger level status
- Transmitter/Receiver FIFO FULL/EMPTY status
- Transmitter/Receiver FIFO empty slots
- whether a Transmit or Receive operation is done or not
- whether End of Frame is received or not
- whether there is any CRC error in received data or not
- whether there is any error in the received data
- whether the controller is busy with transmitting/receiving the data
- whether the VFIR interrupt is generated or not. To generate the interrupt, the corresponding enable bit has to be set in the VFIR Interrupt Enable Register

VFIR Status and Interrupt Identification Register

Offset: 0x4 │ Read/Write: R/W │ Reset: 0xXXXXX0X0 (0bxxxxxxxxxxxxxxxxxxxx0xxx0x000000)

Bit	R/W	Reset	Description
31	RO	X	TX_FTRIG: Transmitter FIFO Trigger level Reached
30	RO	X	TX_FIFO_FULL: Transmitter FIFO is Full
29	RO	X	TX_FIFO_EMPTY: Transmitter FIFO is Empty
28:24	RO	X	TX_FIFO_EMPTY_COUNT: Number of empty slots (words) in the Tx FIFO. The FIFO is a shared Rx/Tx FIFO with a total of decimal 20 entries

Bit	R/W	Reset	Description
23	RO	X	RX_FTRIG: Receiver FIFO Trigger level Reached
22	RO	X	RX_FIFO_FULL: Receiver FIFO is Full
21	RO	X	RX_FIFO_EMPTY: Receiver FIFO is Empty
20:16	RO	X	RX_FIFO_FULL_COUNT: Number of words available to read in the Rx FIFO. The FIFO is a shared Rx/Tx FIFO with a total of decimal 20 entries
15	RO	X	INTR: Global VFIR interrupt (OR of all FIR/VFIR interrupts)
14	RO	X	BUSY: transmitting or receiving data. It is clear when the controller is idle or transmits a Serial infrared Interaction Pulse (SIP)
11	RW	0x0	TX_UNDRN: break. This bit is cleared upon reading from the register.
7	RW	0x0	MISSED_PACKET: Another Rx arrived before the previous was serviced. Serviced is defined as clearing the Rx_EOF flag. This bit is cleared by writing a 1 to this bit.
6	RO	X	BITSYNC_LOCK: Debug status bit which indicates that receiver is Locked to incoming bitstream.
5	RW	0x0	RX_DETECT: Activity detected on Rx pin This bit is cleared by writing a 1 to this bit.
4	RW	0x0	RX_ERR: Receiver Error. This bit is set when an unexpected or illegal data has been received. This can be a break in transmission, illegal chip values or framing errors. This bit is cleared by writing a 1 to this bit.
3	RW	0x0	RX_FOVRN: Receiver FIFO overrun error occurred. This bit is cleared by writing a 1 to this bit.
2	RW	0x0	RX_CRC_ERR: CRC check on input data failed. This bit is cleared by writing a 1 to this bit.
1	RW	0x0	RX_EOF: Received End of Frame. Set when the last byte in frame is within the receiver FIFO. This bit is cleared by writing a 1 to this bit.
0	RW	0x0	DONE: Done flag. Set when controller completes a Transmit or Receive packet, even if the packet finished early due to errors. This bit is cleared by writing a 1 to this bit.

36.5.3 VFIR_IER_0

VFIR Interrupt Enable Register is used to generate VFIR interrupt for different conditions. VFIR Interrupt bit is present in VFIR.

Status and Interrupt Identification Register

For example when the controller finishes the Transmitting/Receiving operation and if IE_DONE is set then Interrupt is generated. Similar is the case for other bits in this register

VFIR Interrupt Enable Register

Offset: 0x8 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxxx0xxxxxxxxxxx0xxx0x000000)

Bit	Reset	Description
31	0x0	IE_TX_FTRIG: Enable Interrupt on Transmitter FIFO Trigger level reached 0 = DISABLE 1 = ENABLE
23	0x0	IE_RX_FTRIG: Enable Interrupt on Receiver FIFO Trigger level reached 0 = DISABLE 1 = ENABLE
11	0x0	IE_TX_UNDRN: transmit a break for the receiving side to abort reception. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x0	IE_MISSED_PACKET: Enable Interrupt for Missed Packet error 0 = DISABLE 1 = ENABLE
5	0x0	IE_RX_DETECT: Enable Interrupt on Activity on Rx Pin 0 = DISABLE 1 = ENABLE
4	0x0	IE_RX_ERR: Enable Interrupt on Receiver Error 0 = DISABLE 1 = ENABLE
3	0x0	IE_RX_FOVRN: Enable Interrupt on Receiver FIFO overrun 0 = DISABLE 1 = ENABLE
2	0x0	IE_RX_CRC: Enable Interrupt for input CRC check failure 0 = DISABLE 1 = ENABLE
1	0x0	IE_RX_EOF: Enable Interrupt for Received End of Frame 0 = DISABLE 1 = ENABLE
0	0x0	IE_DONE: Enable Interrupt for Done Status 0 = DISABLE 1 = ENABLE

36.5.4 VFIR_OFDL_0

VFIR Outgoing Frame Data Length is used to configure the number of bytes of data to be sent.

VFIR Outgoing Frame Data Length

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

Bit	Reset	Description
15:0	0x0	OFDL: Set to the length in bytes of the outgoing frame. Length is calculated on the information field only (address, control and data fields only) without the CRC and flags. When in this mode, the amount of data transferred to the controller is counted and when the count is reached the controller finishes sending the frame in normal fashion by transmitting CRC field, STO flag and then the SIP.

36.5.5 VFIR_IFDL_0

VFIR Incoming Frame Data Length gives the information of the data received in bytes.

VFIR Incoming Frame Data Length

Offset: 0x10 │ Read/Write: RO │ Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	IFDL: Running length in bytes of the incoming data stream. At the end of frame reception when the Received End of Frame interrupt occurs, this register reflects the length of the information field received in that frame, not counting the CRC field and other flags. Resets on the start of next frame reception, when a new STA field has been detected.

36.5.6 VFIR_FIFO_0

The VFIR FIFO is a shared FIFO. It holds the data in Transmitter/Receiver mode.

VFIR.FIFO (VFIR FIFO)

Offset: 0x40 │ Read/Write: R/W │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DATA: Shared FIFO. When transmitting, this FIFO holds the data to be transmitted. When receiving, this FIFO holds the received data.

37.0 SERIAL PERIPHERAL INTERFACE (SPI) CONTROLLER

The Serial Peripheral Interface (SPI) controller is a serial communications link between the processor and on-chip/off-chip serial peripheral devices such as sensors, ADC/DAC devices, and Flash controllers. Tegra® 4 devices support both Master and Slave modes of SPI operation on this interface.

In this section:

- *Word* refers to 32-bit data in the TX or RX FIFO
- *Packet* refers to variable length data that is transmitted or received
  - In Unpacked Mode, packet size can be any length from 1 to 32 bits
  - In Packed Mode, packet size can be 4, 8, 16, or 32 bits

37.1 Functionality

The Tegra 4 SPI Controller works as a Master/Slave on the SPI bus. It has independent transmit and receive FIFOs of 64 x 32 bits each. Software can program the controller to generate transactions of a required packet length on the SPI bus, where a transaction is a sequence of packets in either direction.

The controller supports either APB DMA or program transfer to read and write from the FIFOs as required. At the end of each transaction, an interrupt can be generated, if enabled. Software uses transmit and receive operations in combination with chip select (CS) control to generate commands on the SPI bus.

The interface consists of four signals: Chip Select/Slave Select (CS_N), Clock (SCLK), Master Data Out and Slave Data In (MOSI), and Master Data In and Slave Data Out (MISO).

Features

- Master and Slave functionality
  - Master: All transmission format modes are supported for both transmit and receive operations
  - Slave (transmits): Modes 1 and 3 are supported
  - Slave (receives): All modes are supported
- Independent Rx and Tx FIFOs
- FIFO depth of 64 x 32 bits
- Programmable bit lengths from 3 to 31 bits, resulting in packet sizes of 4 to 32 bits
- PIO or DMA Mode depending on packet size

  Options include Packed/Unpacked, Full-Duplex Mode, Both Enable Bit (x2 Mode), Both Enable Byte (x2 Mode), Bidirectional, Least Significant Bit, Least Significant Byte First (Endian-ness), software or hardware chip-select polarity selection
- Programmable clock phase and polarity
- Programmable delay between consecutive transfers
- Packed mode support for bit lengths of 3 (4-bit packet size), 7 (8-bit packet size), and 15 (16-bit packet size)
- Chip select (CS) can be controlled by software or can be generated automatically by hardware on packet boundaries
- Maximum 4-chip support with programmable CS polarity for each chip select
- DMA support
- Simultaneous receive and transmit operations supported
- The maximum frequency of the device clock is 50 MHz, so the maximum data rate is 50 Mbits/s

37.1.1 Transmission Format

Using the Mode field in the SPI Command1 Register, software sets one of the four modes in which the SPI controller works. The two bits of the Mode field specify the idle (inactive) state of SCLK before the chip select is asserted and the data transmitting/receiving edges of SCLK. Mode [1] determines the SCLK polarity. If the polarity is 0, then the SCLK signal is 0 when idle and transitions to 1 when data transfer starts. If the clock polarity is 1, then the SCLK signal is 1 when idle and transitions to 0 when data transfer starts. Mode [0] is the SCLK phase control bit. If the clock phase is 0, then the receiver latches the data on the first transition of SCLK from the idle state. If the clock phase is 1, then the receiver latches the data on the second transition of SCLK.

Modes supported by the SPI controller are:

- Master: All transmission format modes are supported both for transmitting and receiving
- Slave: Modes 1 and 3 are supported for transmitting
- Slave: All modes are supported for receiving

The following table defines the four modes for the SPI protocol.

Table 123: SPI Modes with Clock Polarity and Phase

SPI Mode	Clock Polarity	Clock Phase	SCLK Inactive State	Data Latch IN	Data Latch OUT
0	0	0	Low	Latched IN on the positive edge of clock	Latched OUT on the negative edge of clock
1	0	1	Low	Latched IN on the negative edge of clock	Latched OUT on the positive edge of clock
2	1	0	High	Latched IN on the negative edge of clock	Latched OUT on the positive edge of clock
3	1	1	High	Latched IN on the positive edge of clock	Latched OUT on the negative edge of clock

37.1.2 Modes of Operation

37.1.2.1 Master and Slave Modes

The SPI controller can be configured to operate as a Master or Slave. When the M/S bit in the SPI Command1 Register is set, Master mode is selected; when the M/S bit in the SPI Command1 Register is cleared, Slave mode is selected. Only the SPI Master can initiate a transaction.

In Master mode, data from the Tx FIFO is transmitted on the MOSI pin. The data from the slave is received on the MISO pin and sent to the Rx FIFO. The Master can simultaneously transmit and receive on MOSI and MISO in Full-Duplex Mode.

In Slave mode, once software enables the SPI controller by setting the PIO bit, it simply waits for the Master to initiate a transaction. Before the transaction begins, the slave logic continuously polls the CS input. When the Master asserts CS and SCLK is transmitted to the Slave, the Slave data is transmitted from the Tx FIFO on MISO and data from MOSI is received in the Rx FIFO. The Slave can also simultaneously transmit and receive on MOSI and MISO in Full-Duplex Mode.

Table 124: Master Mode Port Configuration

Name	Direction	Description
MOSI	Bidirectional	Output in Full-Duplex/Tx Mode Input in Both_Enable Rx Mode
MISO	Bidirectional	Input in Full-Duplex/Rx Mode
SCLK	Output	Output in Both_Enable TX Mode
CS_#x	Output	Slave select signal to x, where x is a number between 0 to 3

Table 125: Slave Mode Port Configuration

Name	Direction	Description
MISO	Bidirectional	Output in Full-Duplex/Tx Mode Input in Both_Enable Rx Mode
MOSI	Bidirectional	Input in Full-Duplex/Rx Mode Output in Both_Enable TX Mode
SCLK	Input	Synchronization clock received from the Master
CS_#	Input	Select signal

When the SPI controller is configured to interface with multiple slaves, the controller has one CS signal for each slave, up to a maximum of four slaves. During a transfer, the master asserts CS specified in the CS_SEL bits in SPI Command1 Register. There can be no more than one slave transmitting data during any particular transfer.

37.1.2.2 DMA and PIO Modes

PIO and DMA are the two main operational modes for the SPI controller. PIO mode is used when the number of packets to the transmitter or receiver is less than or equal to 64. DMA mode is used when the number of packets to the transmitter or receiver is greater than 64. In PIO or DMA mode, the SPI controller sends an interrupt to the processor at the end of each transfer. In this mode software configures the number of packets and number of bits in each packet to be transmitted/received. The limitation of PIO Mode is that it can be used only if the maximum number of packets that can be transferred is less than or equal to 64 packets (the FIFO depth is 64). If software wants to transmit/receive more than 64 packets in PIO Mode, it has to reconfigure the SPI controller every 64 packets.

To send/receive more than 64 packets, the SPI Controller can be configured in DMA mode (enabled by writing 1 to the DMA_EN bit in the SPI DMA Control Register). The SPI controller transmits or receives the number of packets as indicated by the BLOCK_SIZE field in the SPI Block Size Register. BLOCK_SIZE is a 16-bit field, so 2^{16} packets (amounting to 256 KB) can be transmitted/received in a single software configuration.

CS usage guideline: If a hardware-based CS is used, the CS goes to the inactive state after the transfer of packets as programmed in the BLOCK_SIZE register field is completed. If CS has to stay active in multiple consecutive transactions (for example, the first transaction – Tx: 64 Packets; the second transaction – Rx: 128 Packets), then the software based CS has to be used (in hardware-based CS, CS goes to the inactive state after the completion of the first transaction).

37.1.2.3 Packed and Unpacked Modes

SPI communication can be carried out in Packed or Unpacked Mode. These are both present in DMA and PIO Modes.

- Packed Mode: In Packed Mode, data can be transmitted/received with each packet size equal to 4, 8, 16, or 32 bits. The PACKED field and BIT_LEN field in the DMA Command1 Register determine if the mode Packed or Unpacked and the length of each packet. If the BIT_LEN field is set to 03h, each packet is 4 bits long and each word in the Tx/Rx FIFO contains 8 such packets. Similarly, if BIT_LEN is 07h, the packet length is 8. If BIT_LEN is 0Fh, the packet length is 16. If BIT_LEN is 1Fh, the packet length is 32. In Packed Mode, if BIT_LEN is set to 03h and if the number of packets is set to a non-multiple of 4 (for example, 5), the last word in FIFO contains only the fifth packet with 4 valid bits; 28 extra invalid bits will be ignored by the controller for transmits and will contain invalid data for receives.
- Unpacked Mode: If BIT_LEN is set to 03h, then each word in the Tx/Rx FIFO has 4 bits with the remaining bits in each word being ignored. BIT_LEN can be any value from 3 to 31 in Unpacked Mode.

37.1.2.4 Bidirectional Mode

Bidirectional mode is selected when the BIDIR bit is set in the SPI Command1 Register. In this mode, the SPI controller uses only one serial data pin for the interface with external device(s). The M/S bit decides which pin to use. The MOSI pin becomes the serial data I/O (MOMI) pin for the Master Mode, and the MISO pin becomes serial data I/O (SISO) pin for the Slave Mode. The SPI controller does not use the MISO pin in Master Mode and the MOSI pin in Slave Mode.

Table 126: Bidirectional Mode Port Configuration for Master (M/S Bit Set)

Name	Direction	Description
MOSI > MOMI	Bidirectional	Data Output and Input to Slave(s)
MISO > --	--	--
SCLK	Output	Synchronization clock to all Slaves
CS	Output	Slave select signal to x, where x is a number from 0 to 3, inclusive

Table 127: Bidirectional Mode Port Configuration for Slave (M/S Bit Cleared)

Name	Direction	Description
MOSI > --	--	--
MISO > SISO	Bidirectional	Data Output and Input from Slave(s)
SCLK	Input	Synchronization clock
CS	Input	Select signal

37.1.2.5 Both Enable Byte and Bit Modes (SPI x2 Mode)

The Both Enable Byte Mode is selected when the BOTH_EN_BYTE bit is set in the SPI Command1 Register. In this mode, the SPI controller uses both data pins only for transmitting or only for receiving. When the BOTH_EN_BYTE register field is set to 1, the Tx and Rx bits decide whether to transmit or receive. When the Tx bit is set, the MOSI and MISO pins are used to

transmit serial data from the Tx FIFO and Rx FIFO at the same time. When the Rx bit is set, the MISO and MOSI pins are used to receive serial data onto Tx FIFO and Rx FIFO at the same time.

Table 128: Both Enable Mode Port Configuration for Master (Tx Bit Set)

Name	Direction	Description
MOSI	Output	Data Output to Slave(s)
MISO	Output	Data Output to Slave(s)
SCLK	Output	Synchronization clock to all Slaves
CS_N_OUT	Output	Slave select signal to x, where x is a number from 0 to 3, inclusive

Table 129: Both Enable Mode Port Configuration for Slave (Rx Bit Set)

Name	Direction	Description
MOSI	Input	Data Input from Slave(s)
MISO	Input	Data Input from Slave(s)
SCLK	Input	Synchronization clock
CS_N	Input	Select signal

The Both Enable Bit Mode is selected when the BOTH_EN_BIT bit is set in the SPI Command1 Register. This feature is different from Both Enable Byte as follows: In Both Enable Byte Mode, a packet from the Tx FIFO is transmitted or received on the MOSI data line and a packet from the Rx FIFO is transmitted or received on the MISO data line. In Both Enable Bit Mode, a packet from the Tx FIFO is transmitted or received on both MOSI and MISO data lines. Even bits are transmitted or received on the MOSI data line and odd bits are transmitted or received on the MISO data line. This is the same in both Master and Slave Modes.

> **Note**: BOTH_EN_BYTE and BOTH_EN_BIT should not be set to 1 at the same time. BOTH_EN_BYTE or BOTH_EN_BIT should not be set to 1 when the BIDIR bit is set to 1.

37.1.2.6 LSBi_FE and LSBy_FE Modes

> **Note**: LSBi_FE is referred as En_LE_Bit (Enable Little Endian Bit), and LSBy_FE is referred as En_LE_Byte (Enable Little Endian Byte) for better understanding.

These two bits allow a data packet to be transmitted or received in Little Endian or Big Endian format. Once a data packet is read from the Tx FIFO, it undergoes a two-step pre-processing based on En_LE_Byte and En_LE_Bit.

> Step 1. In this step, En_LE_Byte configures whether a packet has to be arranged internally according to the Little Endian byte format or Big Endian byte format. For example, if a 2-byte packet is written into the FIFO by software as Byte 1,Byte 0, setting En_LE_Byte = 0 makes the SPI controller arrange the packet internally as Byte 0,Byte 1(Big Endian format). If En_LE_Byte = 1, the SPI controller internally arranges the packet as Byte 1,Byte 0 (Little Endian format) before the packet is transmitted.

Step 2. In this step, the output of Step 1 is taken, and data is put on the MOSI/MISO line according to En_LE_Bit. If En_LE_Bit is set to 0, the most significant bit of the output of Step 1 is put on the MOSI/MISO line first. If En_LE_Bit is set to 1, the least significant bit of Step 1 output is put on the MOSI/MISO line.

These steps are illustrated in the following table.

Step 1	**FIFO Packet (MSB - LSB)**	**Byte**	**Output of Step 1**
	Byte 1, Byte 0	0	Byte 0, Byte 1
		1	Byte 1, Byte 0
Step 2	**Output of Step 1**	**Bit**	**Data Transmitted on MOSI (Right-most Bit Sent First)**
	Byte 0, Byte 1	0	Byte 1 (0), Byte 1 (1), …, Byte 0 (6), Byte 0 (7)
	Byte 0, Byte 1	1	Byte 0 (7), Byte 0 (6), …, Byte 1 (1), Byte 1 (0)
	Byte 1, Byte 0	0	Byte 0 (0), Byte 0 (1), …, Byte 1 (6), Byte 1 (7)
	Byte 1, Byte 0	1	Byte 1 (7), Byte 1 (6), …, Byte 0 (1), Byte 0 (0)

Similarly for Receive operations, the first packet received will be internally rearranged first based on En_LE_Bit, then the output of the previous step will be rearranged based on En_LE_Byte before writing the final packet into the Rx FIFO.

The following examples explain the effect of these two bits:

- Unpacked Mode
  - Tx: The following example shows how a 20-bit length packet from the FIFO is transmitted. The right-most bit in Column D in the table below is transmitted first. B.7-B.0 indicates that bit 0 is transmitted first followed by bit 1 until bit 7, which is transmitted last.

A	B	C	D
FIFO WORD : 20 (MSB) - 0 (LSB)	**LSBi_FE**	**LSBy_FE**	**Data Tx'ed on MOSI/MISO Line**
	0	0	16, 17, 18, 19, 20, B.8-B-15, B.0 - B.7
	0	1	B.0 - B.7, B.8 - B-15, 16, 17, 18, 19, 20
	1	0	B.7 - B.0, B.15 - B-8, 20, 19, 18, 17, 16
	1	1	B.20 - B.0

  - Rx: The following example shows how a 20-bit length packet from the FIFO is received. The right-most bit in Column D in the table below is received last on the MOSI/MISO lines. B.7-B.0 indicates that bit 7 is received first followed by bit 6 until bit 0, which is received last. B.20 – B.0 are always written into the FIFO after the En_LE_Byte and En_LE_Bit transformations.

A	B	C	D
FIFO WORD : 20 (MSB) - 0 (LSB)	**LSBi_FE**	**LSBy_FE**	**Data Rx'ed on MOSI/MISO Line**
	0	0	B.7 - B.0, B.15 - B.8, 20, 19, 18, 17, 16
	0	1	20, 19, 18, 17, 16, B.15 - B.8, B.7 - B.0
	1	0	16, 17, 18, 19, 20, B.8 - B.15, B.0 - B7
	1	1	B.0 - B.20

- Packed Mode
  - Tx: The following example shows how 3 16-bit packets are transmitted. Column B indicates the valid packets to be transmitted. Because FIFO Word2 is not transmitted entirely, valid packets are only from B.15-

B.0. In Column D, P1(0 -15) indicates Packet1 of Column B is transmitted, B.15 is transmitted first, and B.0 is transmitted last.

A	B : Valid Packets	B	C	D
FIFO WORD1 : 31 (MSB) - 0 (LSB)	**P1(15-0), P0(15-0)**	**LSBi_FE**	**LSBy_FE**	**Data Tx'ed on MOSI/MISO Line**
FIFO WORD2 : 31 (MSB) - 0 (LSB)	**P2(15-0)**	0	0	P2(8-15), P2(0-7), P1(8-15), P1(0-7), P0(8-15), P0(0-7)
		0	1	P2(0-7), P2(8-15), P1(0-7), P1(8-15), P0(0-7), P0(8-15)
		1	0	P2(7-0), P2(15-8), P1(7-0), P1(15-8), P0(7-0), P0(15-8)
		1	1	P2(15-8), P2(7-0), P1(15-8), P1(7-0), P0(15-8), P0(7-0)

- Rx: The following example shows how 3 16-bit packets are received. Column B indicates the valid received packets in the Rx FIFO. In column D, P1(0 -15) indicates Packet1 of Column B is received, B.0 is received first, and B.15 is received last.

A	B : Valid Packets	B	C	D
FIFO WORD1 : 31 (MSB) - 0 (LSB)	**P1(15-0), P0(15-0)**	**LSBi_FE**	**LSBy_FE**	**Data Rx'ed on MOSI/MISO Line**
FIFO WORD2 : 31 (MSB) - 0 (LSB)	**P2(15-0)**	0	0	P0(7-0), P0(15-8), P1(7-0), P1(15-8), P2(7-0), P2(15-8)
		0	1	P0(15-8), P0(7-0), P1(15-8), P1(7-0), P2(15-8), P2(7-0)
		1	0	P0(8-15), P0(0-7), P1(8-15), P1(0-7), P2(8-15), P2(0-7)
		1	1	P0(15-8), P0(7-0), P1(15-8), P1(7-0), P2(15-8), P2(7-0)

37.2 SPI Programming Guidelines

There are two basic modes of operation: DMA and PIO modes. It is required that software sets up all parameters in the SPI Command1 and Command2 registers, SPI CS Timing 1 and 2 registers, SPI FIFO Control/Status register, SPI DMA Control register, and SPI Block Size register before enabling the PIO bit in any of these modes.

37.2.1 DMA Mode

This mode is enabled by writing 1 to the DMA bit in the SPI DMA Control register. In this mode, the SPI controller transmits or receives the number of packets as indicated by the BLOCK_SIZE field in the SPI Block Size register.

If the PACKED bit is set and BIT_LEN is set to 7, then all FIFO words contain 4 packets to transfer (transmit or receive). Packets will be transferred as per the LSBi_FE and LSBy_FE bit configurations (see the LSBi_FE and LSBy_FE Modes section), with packet 0 in byte 0 of the FIFO and packet 3 in byte 3 of the FIFO.

In Unpacked Mode, if BIT_LEN is set to N, each packet will consist of (N + 1) bits. These bits will be transmitted/received in the Tx_FIFO/Rx_FIFO as per the LSBi_FE and LSBy_FE bit configurations (see the LSBi_FE and LSBy_FE Modes section). Any remaining bits in the FIFO will be ignored by the hardware. The maximum packet length is 32, which can be selected by setting BIT_LEN to 31. In this case, all data bits in the FIFO contain valid packet data.

A DMA request will be generated to APB_DMA in this mode depending on the setting of Tx_TRIG and Rx_TRIG. If transmits are enabled, setting Tx_TRIG to 00 will generate a Tx DMA request whenever the Tx FIFO has one word of space available (if not full). Setting Tx_TRIG to 01 will generate a Tx DMA request whenever the Tx FIFO has 4 words of space available. If receives are enabled, setting Rx_TRIG to 00 will generate an Rx DMA request whenever the Rx FIFO has one word of data available (is not empty). Setting Rx_TRIG to 01 will generate an Rx DMA request whenever the Rx FIFO has 4 words of data available.

37.2.2 PIO Mode

This mode is enabled by writing 1 to the PIO bit in the SPI Command1 register. In this mode, the SPI controller transmits/ receives as many packets as configured by software in the SPI Block Size Register.

PIO Mode has the same features as DMA Mode. The difference between PIO Mode and DMA Mode is that in PIO Modek the maximum number of packets that can be transmitted or received is less than or equal to 64.

37.2.3 Interrupt Generation

The SPI controller generates an interrupt to the processor at the end of a transfer in PIO Mode or when an error is detected (both in PIO and DMA Modes) if the IE.TX or IE.RX bit in the SPI DMA Control register is enabled for transmit and receive modes of operation, respectively.

If IE.TX is enabled during transmits, an interrupt is generated whenever RDY or one of the FIFO status bits in the SPI FIFO Control/Status Register is set by the hardware. During transmits, when the SPI controller is configured as a Slave, if software/APB DMA cannot fill the transmit FIFO fast enough, hardware will set the Tx_FIFO_UNR bit and an underrun condition is generated. If software tries to write to a full Tx FIFO, hardware sets the Tx_FIFO_OVF bit as an indication that software attempted to overflow the Tx FIFO. Hardware makes sure that the overflowing data is never written to the Tx FIFO.

If IE.RX is enabled during receives, an interrupt is generated whenever RDY or one of the status bits in the SPI FIFO Control/Status Register is set by the hardware. During receives, when the SPI controller is configured as a Slave, if software/APB DMA cannot read the receive FIFO fast enough, hardware will set the Rx_FIFO_OVF bit and an overflow condition is generated. However, if software tries to read from an empty Rx FIFO, hardware sets the RXF_UNR bit as an indication that software attempted to underrun the Rx FIFO.

The interrupt can be cleared by writing a 1 to the source of the interrupt. If the interrupt is generated by assertion of RDY, then writing a 1 to the RDY bit clears the interrupt. If the interrupt is generated by assertion of Tx_FIFO_OVF, then writing a 1 to the TXF_OVF bit clears the interrupt. If the interrupt is generated by assertion of Rx_FIFO_UNR, then writing a 1 to RXF_UNR bit clears the interrupt.

> **Guideline for Automotive use cases**: In DMA Mode, if interrupts are enabled, the SPI software driver has to deal with 3 interrupts: one from the SPI controller and two from the Tx and Rx APB DMA channels. This might make it complicated for the driver and also stresses the system. In such cases, software can use the Rx DMA interrupt as the final one at which point all hardware activities can be assumed to be complete. But, in variable transfer lengths (continuous mode), software has to rely on the controller's interrupts.

37.2.4 Clock Initialization and Control

The SPI controller runs at 50 MHz at its interface to external SPI devices. The internal SPI controller clock should run at the same frequency as the outgoing Sclk_Out in Master Mode. In Slave Mode, the internal SPI interface clock frequency has to be 50% greater than the external clock (Sclk_in).

The SPI clock is selected from different clock sources using a generic clock switch module. The clock switch module supports glitch-free switching of clock from one source to another. There are 4 clock sources:

- osc_clk
- pllP_out
- pllC_out
- pllM_out0

The selected clock is divided by an 8-bit value written in the divider register. The clock is then trimmed, and gated branches are generated for the SPI instances.

The CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0, CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0, and CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0 registers in the CAR (Clock and Reset) block contain the enable bits for the SPI controller. In addition, the clock source and divider registers (CLK_SOURCE_SBC1, CLK_SOURCE_SBC2, CLK_SOURCE_SBC3, CLK_SOURCE_SBC4, CLK_SOURCE_SBC5, and CLK_SOURCE_SBC6) contain the 2-bit field SBCx_CLK_SRC to determine the PLL clock source, while the SBCx_CLK_DIVISOR field determines the divide ratio.

See the Clock and Reset Controller section for more information on clock configuration.

Controller Reset

The SPI controller can be reset by writing 1 to SPI1 (bit 11) in the RST_DEVICES.H (0x6000:6008) register. Software needs to write a 0 to this bit to bring the SPI controller out of reset.

See the Clock and Reset Controller section for more information on SPI controller reset.

Slave Mode GPIO Synchronization

Because SPI does not support flow control, the GPIO is required to have the proper communication to avoid any clock loss from the Master. GPIO can be used in the following way for proper synchronization:

- SPI slave client software will write to the Config registers of the SPI controller and make the SPI slave ready.
- After that the slave client software will toggle GPIO to inform the Master that the Slave is ready.
- Then the SPI master sends SCLK to the slave device.

If there is no GPIO to tell the Master that the Slave is ready, there is a chance to lose clock/data.

Guidelines

This section provides guidelines for programming the SPI controller:

- First program all the required register fields (no particular order is required except for the PIO/DMA bit, which has to be programmed last):
  - Clock Mode
  - Packed/Unpacked Mode
  - Tx_EN/Rx_EN
  - BOTH_EN_BYTE/BOTH_EN_BIT
  - LSBi_FE/LSBy_FE
  - BIT_LEN and BLOCK_SIZE values
  - CS_SW_HW (software based or hardware based)
    - If CS_SW_HW is set, the value on CS_SW_VAL will be driven out.
    - When CS HW is used program the Setup & Hold Values in CS Timing Register.
  - Program the DMA Trig values appropriately, if DMA Mode is used
  - Enable interrupts if PIO Mode is used
- Lastly, program PIO or DMA to indicate the start of transfer.

Once software enables PIO/DMA, no required bits can be changed until the end of transfer except for PIO/DMA bit. Clearing the PIO/DMA bit will end the transaction. In case the SPI controller is configured to Receive Mode and software clears the PIO/DMA bit, then the partial data which the controller received until then will be written into the FIFO. This is true both for Master and Slave.

Do not enable conflicting features. For example, avoid enabling BOTH_EN_BIT (or) BOTH_EN_BYTE along with bidirectional (BIDIR) mode.

In Slave Mode after PIO or DMA is set, hardware first waits for the CS to go low by one of the 4 Masters. Then on reception of Sclk, transfer begins. If CS or Sclk is removed by the Master in the middle of a transfer, software must terminate the transaction by clearing the PIO/DMA bit. In case of Rx, hardware then writes the last received packet (incomplete packet) to the Rx FIFO. If software does not clear the PIO/DMA bit, the SPI controller will just keep waiting for the Sclk from the Master. If Sclk resumes later, the transaction continues from where it paused earlier.

If any error conditions occur, hardware will set the corresponding status bits in the SPI FIFO Control/Status register and stop the transfer. If IE is enabled, an interrupt is generated. Software has to clear the source of the interrupt by writing a 1 to it, after the completion of ISR.

Error Conditions

When the SPI controller is configured as a Master, the following error scenarios can happen:

- Tx_FIFO overflow

  Tx_FIFO overflow happens when the CPU/APB_DMA writes into the Tx FIFO when it is full. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the Tx_FIFO_OVF bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Tx_FIFO_OVF bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.
- Rx_FIFO overflow

  Rx_FIFO overflow happens when the CPU/APB_DMA writes into the Rx FIFO when it is full in BOTH_EN Mode. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with Rx_FIFO_OVF bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Rx_FIFO_OVF bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.
- Tx_FIFO underrun

  Tx_FIFO underrun happens when the CPU/APB_DMA reads from the Tx FIFO when it is empty. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the Tx_FIFO_UNR bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Tx_FIFO_UNR bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.
- Rx_FIFO underrun

  Rx_FIFO underrun happens when the CPU/APB_DMA reads from the Rx FIFO in BOTH_EN Mode. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the Rx_FIFO_UNR bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and the Rx_FIFO_UNR bit, along with the Interrupt bit. To start a new transaction, software has to flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

  > **Note**: In Master Mode, all the errors above can happen only when the CPU/APB_DMA reads/writes an empty/full FIFO. The SPI controller will never write/read a full/empty FIFO. Instead the controller will pause (stops sending clocks to the slave with chip select being in an active state) whenever the FIFOs are full/empty.

When the SPI controller is configured as a Slave, the following error scenarios can happen:

- Tx_FIFO overflow

  Tx_FIFO overflow happens when the CPU/APB_DMA or the SPI controller writes into the Tx FIFO when it is full. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with Tx_FIFO_OVF bit in the status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and Tx_FIFO_OVF bit. To start a new transaction, software has to disable the Master from sending clocks and then flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Rx_FIFO overflow

  Rx_FIFO overflow happens when the CPU/APB_DMA or the SPI controller writes into Rx FIFO when it is full in BOTH_EN Mode. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with Rx_FIFO_OVF bit in the status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and Rx_FIFO_OVF bit along with the Interrupt bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Tx_FIFO underrun

  Tx_FIFO underrun happens when the CPU/APB_DMA or the SPI controller reads from Tx FIFO when it is empty. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the TX_FIFO_UNR bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and TX_FIFO_UNR bit along with the Interrupt bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Rx_FIFO underrun

  Rx_FIFO underrun happens when the CPU/APB_DMA or the SPI controller reads from the Rx FIFO in BOTH_EN Mode. In this case, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and flag the error bit along with the RX_FIFO_UNR bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the error bit and RX_FIFO_UNR bit along with the Interrupt bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- CS_inactive

  When the SPI controller is configured as a Slave and cs_active between packets is set, if an external Master deasserts the chip select in the middle of a transaction, the SPI controller will end the transaction by setting the PIO/DMA_EN bit to 0, and sets the CS_INACTIVE bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the CS_INACTIVE bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register.

- Frame_End

  When the SPI controller is configured as a Slave and DMA-continuous and cs_active between packets is set, if an external Master stops sending clocks for more than slave_idle_clock_count programmed in the register, the SPI controller will end the transaction by setting the DMA bit to 0, and sets the FRAME_END bit in the SPI FIFO Control/Status register. The SPI controller will generate an interrupt if interrupts are enabled. Software has to clear the FRAME_END bit along with the Interrupt bit. To start a new transaction, software has to disable the Master from sending clocks and flush the FIFOs by writing to the TX_FIFO_FLUSH/RX_FIFO_FLUSH bits in the SPI FIFO Control/Status register. If the Master resumes the clk within “slave_idle_clock_count”, the SPI controller will also pause and resume the transaction whenever ext_clk is available.

Programming Trimmers

The 3 programmable trimmers in the SPI controller are used only in Master Mode:

- SPI-Tx trimmer

  This trimmer is used in Master Tx mode to adjust/center the outgoing data with respect to the outgoing clock.

- SPI-Rx trimmer

  This trimmer is used in Master Rx mode to delay the loopback clock to the Rx shift registers. This trimmer is used to adjust the Master SCLK with respect to the SPI slave device’s Tx data.

- SPI-Spare Control Register Byte1

This 3-bit trimmer controls the internal timing of the Master mode Rx datapath. This trimmer along with the SPI Rx trimmer may be set independently to adjust the loopback clock delay. This trimmer adjusts the programmable delay on the u_spi_slave_rx* flops in inbound paths. These are half cycle paths, and delays have been added to increase Setup margin on these paths.

- The SPI controller has a programmable delay chain with a 3-tap delay chain. The 3 taps use 100D, 100D, and 60D and are programmed as follows:
  - 000: Bypass through each mux: will see 3 muxes in path
  - 001: 60D
  - 010: 100D
  - 100: 100D
  - 011: 60D + 100D
  - 110: 100D + 100D
  - 101: 100D+60D
  - 111: 100D+100D+60D
- Spare cells used as control for these delay taps:
  - (u_spi1_shift_padmacro/spi_spare_control_byte1_in_reg_2_,
  - u_spi1_shift_padmacro/spi_spare_control_byte1_in_reg_1_,
  - u_spi1_shift_padmacro/spi_spare_control_byte1_in_reg_0_)

37.3 SPI Controller Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

There are six sets of the SPI controller registers, one for each of the six SPI controllers.

37.3.1 SPI_COMMAND1

SPI Command1 Register

This register is used to set the bit length and to select the transfer mode.

Chip Select can also be selected to be in hardware mode as software mode with both active high and active low polarities to support devices with varying CS polarities.

Offset: 0x0 │ Read/Write: R/W │ Reset: 0x43d00000 (0b01000011110100000000000xxxxx000000)

Bit	Reset	Description
31	0x0	PIO: Program 1 after all the other bits in the SPI_COMMAND2 and SPI_COMMAND1 registers are programmed to start the transfer. Hardware clears this bit automatically after the transfer is done. Clearing of this bit by software will stop the shifter and latch the partial data into the buffer (in Receive Mode). 0 = STOP (default) 1 = GO
30	0x1	M/S: Master/Slave mode select. 0 = Slave Mode (external clock) 1 = Master Mode (internal clock) (default)
29:28	0x0	MODE: The SPI interface clock mode has to be programmed according to the device with which it is communicating. 0 = Mode 0 (default) 1 = Mode 1 2 = Mode 2

Bit	Reset	Description
		3 = Mode 3 Master: All modes are supported both for Tx and Rx. Slave: Modes 1and 3 are supported for Tx Slave: All modes are supported for Rx.
27:26	0x0	CS_SEL: In Master Mode, these bits are programmed to select a slave in the multi-slave environment. 0 = Selects CS0 (default) 1 = Selects CS1 2 = Selects CS2 3 = Selects CS3
25	0x1	CS_POL_INACTIVE#3: In Master or Slave Mode, the inactive value of the external device's cs value, which is connected to CS3, needs to be programmed. 1 = CS3 Inactive value of External device is high, 0 = CS3 Inactive value of External device is low 0 = LOW 1 = HIGH (default)
24	0x1	CS_POL_INACTIVE#2: In Master or Slave Mode, the inactive value of the external device's cs value, which is connected to CS2, needs to be programmed, 1 = CS2 Inactive value of External device is high, 0 = CS2 Inactive value of External device is low 0 = LOW 1 = HIGH (default)
23	0x1	CS_POL_INACTIVE#1: In Master or Slave Mode, the inactive value of the external device's cs value, which is connected to CS1, needs to be programmed. 1 = CS1 Inactive value of External device is high, 0 = CS1 Inactive value of External device is low 0 = LOW 1 = HIGH (default)
22	0x1	CS_POL_INACTIVE#0: In Master or Slave Mode, the inactive value of the external device's cs value, which is connected to CS0, needs to be programmed. 1 = CS0 Inactive value of External device is high, 0 = CS0 Inactive value of External device is low 0 = LOW 1 = HIGH (default)
21	0x0	CS_SW_HW: Software control of the SPI_CS signal in Master Mode. In Slave Mode, this bit need not be programmed. 1 = CS controlled by software; 0 = CS controlled by hardware 0 = SPI _CS#(CS_SEL) is driven to the active state during packet transfers by the hardware (default) 1 = SPI _CS#(CS_SEL) is driven with the value in the CS_SW_VAL bit
20	0x1	CS_SW_VAL: CS signal value in Master Mode. In Slave Mode, this bit need not be programmed. If CS_SW_HW is 1, then the value in CS_SW_VAL is driven out on CS. If CS_SW_HW is 0, then this bit has no effect. 0 = CS is LOW 1 = CS is HIGH (default)
19:18	0x0	IDLE.SDA: Inactive data signal format. Controls the output enable of the SDA line when the controller is inactive and not doing data transfers. 0 = Drive low. (Master Mode) (default) 1 = Drive high. (Master Mode) 2 = External Pull Down. (Slave Mode) 3 = External Pull high. (Slave Mode)
17	0x0	BIDIR: Bidirectional Transfer Control Bit: This bit enables the bidirectional mode of the transfer. 0 = Normal Mode (default) 1 = Bidirectional Mode
16	0x0	LSBi_FE: Least Significant Bit First Enable. This bit works in combination with the LSBy_FE bit. Refer to the LSBi_FE and LSBy_FE Modes section for the encodings. The default is 0.
15	0x0	LSBy_FE: Least Significant Byte First Enable. This bit works in combination with the LSBi_FE bit. Refer to the LSBi_FE and LSBy_FE Modes section for the encodings. The default is 0.
14	0x0	BOTH_EN_BIT (x2 Mode): Both MISO and MOSI can also be used to transmit or receive at the same time when this bit is set. Even bits from the Tx FIFO packet are

Bit	Reset	Description
		transmitted/received on MOSI, and Odd bits from the Tx FIFO packet are transmitted/received on MISO. Refer to the Both Enable Byte and Bit Modes (SPI x2 Mode) section for more information. 0 = Normal Mode (default) 1 = Transmit/Receive on both MISO and MOSI at the same time when the Tx_EN and Rx_EN bits are set
13	0x0	BOTH_EN_BYTE (x2 Mode): The 2 FIFOs can also be used to transmit or receive at the same time when this bit is set. Refer to the Both Enable Byte and Bit Modes (SPI x2 Mode) section for more information. **Note**: the BIDIR bit has to be 0 before this bit is set. 0 = Normal Mode (default) 1 = Transmit/Receive on both MISO and MOSI at the same time when the Tx_EN and Rx_EN bits are set
12	0x0	Rx_EN: Receive enable. Setting this bit to 1 enables the controller to receive the data. 0 = DISABLE (default) 1 = ENABLE
11	0x0	Tx_EN: Transmit enable. Setting this bit to 1 enables the controller to transmit the data. 0 = DISABLE (default) 1 = ENABLE
10:6	N/A	Reserved for future use. Always write zeros.
5	0x0	PACKED: Packed mode enable bit.. 0 = Unpacked Mode (default) 1 = Packed Mode. This is only valid if the BIT_LEN field is set to either 3 (4-bit transfer), 7 (8-bit transfer), 15 (16-bit transfer), or 31 (32-bit transfer).
4:0	0x0	BIT_LEN: This field is used during PIO and DMA transfers. It represents the number of bits transmitted/received in either Packed or Unpacked Mode. The minimum bit length in Master Mode is 4 and the minimum length in Slave Mode is 1 byte (BIT_LEN = 7). The default is 0. 3 = 4-bit transfer N = N+1-bit transfer 31 = 32-bit transfer

37.3.2 SPI_COMMAND2

SPI Command2 Register

Offset: 0x4 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx000000000000)

Bit	Reset	Description
31:12	N/A	Reserved for future use. Always write zeros.
11:6	0x0	Tx_Clk_TAP_DELAY: Delays the clock going out to the external device with these tap values. Useful only in Master Mode. The default is 0.
5:0	0x0	Rx_Clk_TAP_DELAY: Delays the clock coming in from the external device with these tap values. Useful only in Master Mode. The default is 0. In Slave Mode, this value should always be 0.

37.3.3 SPI_CS_TIM1

SPI CS Timing1 Register

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

Bit	Reset	Description
31:28	0x0	CS_SETUP_TIME#3: Specifies the setup time of the chip select to the data being transmitted or received on CS#3 in numbers of cycles (16 cycles maximum). The default is 0.

Bit	Reset	Description
		1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
27:24	0x0	CS_HOLD_ TIME#3: Specifies the hold time of the chip select to the data being transmitted or received on CS#3 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
23:20	0x0	CS_SETUP_ TIME#2: Specifies the setup time of the chip select to the data being transmitted or received on CS#2 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
19:16	0x0	CS_HOLD_ TIME#2: Specifies the hold time of the chip select to the data being transmitted or received on CS#2 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
15:12	0x0	CS_SETUP_ TIME#1: Specifies the setup time of the chip select to the data being transmitted or received on CS#1 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
11:8	0x0	CS_HOLD_ TIME#1: Specifies the hold time of the chip select to the data being transmitted or received on CS#1 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
7:4	0x0	CS_SETUP_ TIME#0: Specifies the setup time of the chip select to the data being transmitted or received on CS#0 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
3:0	0x0	CS_HOLD_ TIME#0: Specifies the hold time of the chip select to the data being transmitted or received on CS#0 in numbers of cycles (16 cycles maximum). The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.

37.3.4 SPI_CS_TIM2

SPI CS Timing2 Register

Offset: 0xc │ Read/Write: R/W │ Reset: 0x20202020 (0bxx100000xx100000xx100000xx100000)

Bit	Reset	Description
31:30	N/A	Reserved for future use. Always write zeros.
29	0x1	CS_ACTIVE_BETWEEN_PACKETS#3: Specifies if CS stays active between two packets on CS#3 1 = CS active between two packets (default) 0 = CS inactive between two packets
28:24	0x0	CYCLES_BETWEEN_PACKETS#3: Specifies the number of cycles between two packets in the PIO/DMA Mode for communication on CS#3. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
23:22	N/A	Reserved for future use. Always write zeros.

Bit	Reset	Description
21	0x1	CS_ACTIVE_BETWEEN_PACKETS#2: Specifies if CS stays active between two packets on CS#2 1 = CS active between two packets (default) 0 = CS inactive between two packets
20:16	0x0	CYCLES_BETWEEN_PACKETS#2: Specifies the number of cycles between two packets in the PIO/DMA Mode for communication on CS#2. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
15:14	N/A	Reserved for future use. Always write zeros.
13	0x1	CS_ACTIVE_BETWEEN_PACKETS#1: Specifies if CS stays active between two packets on CS#1 1 = CS active between two packets (default) 0 = CS inactive between two packets
12:8	0x0	CYCLES_BETWEEN_PACKETS#1: Specifies the number of cycles between two packets in the PIO/DMA Mode for communication on CS#1. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.
7:6	N/A	Reserved for future use. Always write zeros.
5	0x1	CS_ACTIVE_BETWEEN_PACKETS#0: Specifies if CS stays active between two packets on CS#0 1 = CS active between two packets (default) 0 = CS inactive between two packets
4:0	0x0	CYCLES_BETWEEN_PACKETS#0: Specifies the number of cycles between two packets in the PIO/DMA Mode for communication on CS#0. The default is 0. 1 = 2 clock cycle delay 2 = 3 clock cycle delay, etc.

37.3.5 SPI_TRANS_STATUS

SPI TRANSFER Status Register

Read this register for the status of the transfer.

Offset: 0x10 │ Read/Write: R/W │ Reset: 0x00ff0000 (0bx0xxxxx11111111xxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
31	N/A		Reserved for future use. Always write zeros.
30	RW	0x0	RDY: Ready bit. This bit is set to 1 at the end of every transfer, and an interrupt is also generated if the corresponding interrupt enable is set in PIO/DMA Mode. Software writes a 1 to clear it. The interrupt is also cleared when this bit is cleared. 0 = NOT_READY (default) 1 = READY
29:24	N/A		Reserved for future use. Always write zeros.
23:16	RW	0xff	SLV_IDLE_COUNT: Slave Continuous Mode. If Sclk is not received for this number of cycles, then Slave Continuous mode is terminated, and the status bit FRAME_END is set. The default is 255 (8'hff).
15:0	RO	X	BLOCK_COUNT: Counts the number of packets in a transaction (Tx or Rx) in DMA/PIO Mode. In DMA Continuous Mode, this field gives the number of packets only if the number of packet is less than 2^{16}.

37.3.6 SPI_FIFO_STATUS

SPI Control/Status FIFO Status Register

SPI STATUS register: Read this register to know the status of the transfer. Error bit is set whenever we encounter Errors such as Underflow/Overflow

Offset: 0x14 | Read/Write: R/W | Reset: 0x00400005 (0b00xxxxxxxxxxxxxx00xxxxx00000xxxx)

Bit	R/W	Reset	Description
31	RW	0x0	CS_INACTIVE: When the SPI is in Slave Mode, if CS is deasserted, this bit is set to indicate an error and the received number of bits is written in the Rx FIFO. An interrupt is generated if IE.Tx or IE.Rx is set. The default is 0.
30	RW	0x0	FRAME_END: When the SPI is in Slave Continuous mode, if Sclk is not received for the number of clocks specified in the SLV_IDLE_COUNT field, the continuous mode is terminated and this status bit is set. The default is 0.
29:23	RO	X	RX_FIFO_FULL_COUNT: Indicates the number of slots in the receive FIFO remaining before the FIFO is empty. This field is used by software for debugging purposes.
22:16	RO	X	TX_FIFO_EMPTY_COUNT: Indicates the number of slots in the transmit FIFO remaining before the FIFO is full. This field is used by software for debugging purposes.
15	RW	0x0	RX_FIFO_FLUSH: Software writes a 1 to this bit to flush the Rx FIFO. This bit reads as 1 when the flush operation is in progress and returns to 0 when it is finished. 0 = NOP (default) 1 = FLUSH
14	RW	0x0	TX_FIFO_FLUSH: Software writes a 1 to this bit to flush the Tx FIFO. This bit reads as 1 when the flush operation is in progress and returns to 0 when it is finished. 0 = NOP (default) 1 = FLUSH
13:9	N/A		Reserved for future use. Always write zeros.
8	RW	0x0	ERR: Will be set to 1 by hardware when errors such as underflow/overflow occur. Software writes a 1 to clear the flag. 0 = OK (default) 1 = ERROR
7	RW	0x0	TX_FIFO_OVF: TX FIFO Overflow. This bit is set to 1 whenever the SPI controller or software tries to write to a full Tx FIFO. An interrupt is generated if the interrupt enable is set for transmit operations (IE.TX in the SPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
6	RW	0x0	TX_FIFO_UNR: TX FIFO Underrun. This bit is set to 1 whenever the SPI controller or software tries to read from an empty Tx FIFO. An interrupt is generated if the interrupt enable is set for transmit operations (IE.TXC in the SPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
5	RW	0x0	RX_FIFO_OVF: RX FIFO Overflow. This bit is set to 1 whenever the SPI controller or software tries to write into a full Rx FIFO. An interrupt is generated if the interrupt enable is set for receive operations (IE.RXC in the SPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR
4	RW	0x0	RX_FIFO_UNR: RX FIFO Underrun. This bit is set to1 whenever the SPI controller or software tries to read from an empty Rx FIFO. An interrupt is generated if the interrupt enable is set for receive operations (IE.RXC in the SPI_DMA_CTL register). Software writes a 1 to clear this bit. 0 = OK (default) 1 = ERROR

Bit	R/W	Reset	Description
3	RO	X	TX_FIFO_FULL: TX FIFO Full Status. Hardware sets this bit to 1 if the Tx FIFO is full; otherwise this bit is 0. The Tx FIFO is empty at power on reset (POR). 0 = NOT_FULL (default) 1 = FULL
2	RO	X	TX_FIFO_EMPTY: TX FIFO Empty Status. Hardware sets this bit to 1 if the Tx FIFO is empty; otherwise this bit is 0. The Tx FIFO is empty at POR. 0 = NOT_EMPTY 1 = EMPTY (default)
1	RO	X	RX_FIFO_FULL: RX FIFO Full Status. Hardware sets this bit to 1 if the Rx FIFO is full; otherwise this bit is 0. The Rx FIFO is empty at POR. 0 = NOT_FULL (default) 1 = FULL
0	RO	X	RX_FIFO_EMPTY: RX FIFO Empty Status. Hardware sets this bit to 1 if the Rx FIFO is empty; otherwise this bit is 0. The Rx FIFO is empty at POR. 0 = NOT_EMPTY 1 = EMPTY (default)

37.3.7 SPI_TX_DATA

SPI Transmit Data Register

Offset: 0x18 │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SPI_Tx_DATA. Transmit Data. This register holds the last data that was transmitted by the SPI controller.

37.3.8 SPI_RX_DATA

SPI Receive Data Register

Offset: 0x1c │ Read/Write: RO │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SPI_Rx_DATA. Receive Data. This register holds the last data that was received by the SPI controller.

37.3.9 SPI_DMA_CTL

SPI DMA Control Register

SPI DMA Control Register: Used in the DMA transfers. We can set trigger levels in this register

Offset: 0x20 │ Read/Write: R/W │ Reset: 0x00000000 (0b0000xxxxxxx00xx00xxxxxxxxxxxxxxx)

Bit	Reset	Description
31	0x0	DMA: Enable DMA Mode transfer. Software writes a 1 to this bit to start a transfer in the DMA mode. All fields in the SPI_COMMAND1 and SPI_DMA_CTL register must be set before writing a 1 to this bit. This bit is cleared by the SPI controller after all packets have been transferred as indicated by the DMA_BLOCK_SIZE field. Clearing this bit by software will stop the shifter and latch the partial data into buffer. 0 = DMA Mode is disabled (default) 1 = DMA Mode is enabled
30	0x0	CONT: Enable Continuous Mode transfer. 0 = Continuous Mode is disabled (default) 1 = Continuous Mode is enabled

Bit	Reset	Description
29	0x0	IE.RX: Interrupt enable on receive completion. 0 = Disable interrupt generation for receives (default) 1 = Enable interrupt generation at the end of a receive transfer
28	0x0	IE.TX: Interrupt enable on transmit completion. 0 = Disable interrupt generation for transmits (default) 1 = Enable interrupt generation at the end of a transmit transfer
27:21	N/A	Reserved for future use. Always write zeros.
20:19	0x0	RX_TRIG: Receive FIFO trigger level. 00: 1 word. DMA trigger is asserted whenever there is at least 1 packet in the RX FIFO. (default) 01: 4 words. DMA trigger is asserted when there are at least 4 packets in the RX FIFO. 10: 8 words. DMA trigger is asserted when there are at least 8 packets in the RX FIFO. 11: 16 words. DMA trigger is asserted when there are at least 16 packets in the RX FIFO. (Presently, the APB DMA does not support this trigger level.)
18:17	N/A	Reserved for future use. Always write zeros.
16:15	0x0	TX_TRIG: Transmit FIFO trigger level. 00: 1 word. DMA trigger is asserted whenever there is space for at least 1 packet in the TX FIFO. (default) 01: 4 words. DMA trigger is asserted when there is space for at least 4 packets in the TX FIFO. 10: 8 words. DMA trigger is asserted when there is space for at least 8 packets in the TX FIFO. 11: 16 words. DMA trigger is asserted when there is space for at least 16 packets in the TX FIFO. (Presently, the APB DMA does not support this trigger level.)

37.3.10 SPI_DMA_BLK

SPI Block Size Register

Offset: 0x24 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31:16	N/A	Reserved for future use.
15:0	0x0	BLOCK_SIZE: Size of data block to be transferred in PIO/DMA Mode. The default is 0. In DMA Mode, the maximum number of 32-bit packets that can be transferred is 2^{16} = 65536. In PIO Mode, the maximum number of 32-bit packets that can be transferred is 64 (FIFO depth). The Block Size has to be programmed 1 packet lower than intended; for example, a value of 3 indicates 4 packets are to be transferred.

37.3.11 SPI_FIFO1

SPI TX FIFO Register

Offset: 0x108 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SPI Tx_FIFO. Tx FIFO.

37.3.12 SPI_FIFO2

SPI RX FIFO Register

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	SPI Rx_FIFO. Rx FIFO.

37.3.13 SPI_SPARE_CTLR

SPI Spare Control Register

Offset: 0x18c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:11	0x0	Reserved for future use, always write zero.
10:8	0x0	SPI_SPARE_CONTROL_REGISTER_BYTE2. Program these bits with Rx_Clk_TAP_DELAY in the COMMAND2 register to adjust the clock delay on internal registers. Useful in Master Mode only. The default is 0. In Slave Mode, this value should always be 0.
7:0	0x0	Reserved for future use, always write zero.

38.0 ONE WIRE BATTERY CONTROLLER

The One Wire Controller (OWR) implements a device communications bus system that provides low-speed data, signaling, and power over a single signal. The OWR interface uses two wires for this – one for ground and the other for power and data. (However, the ground can be shared with power delivery.)

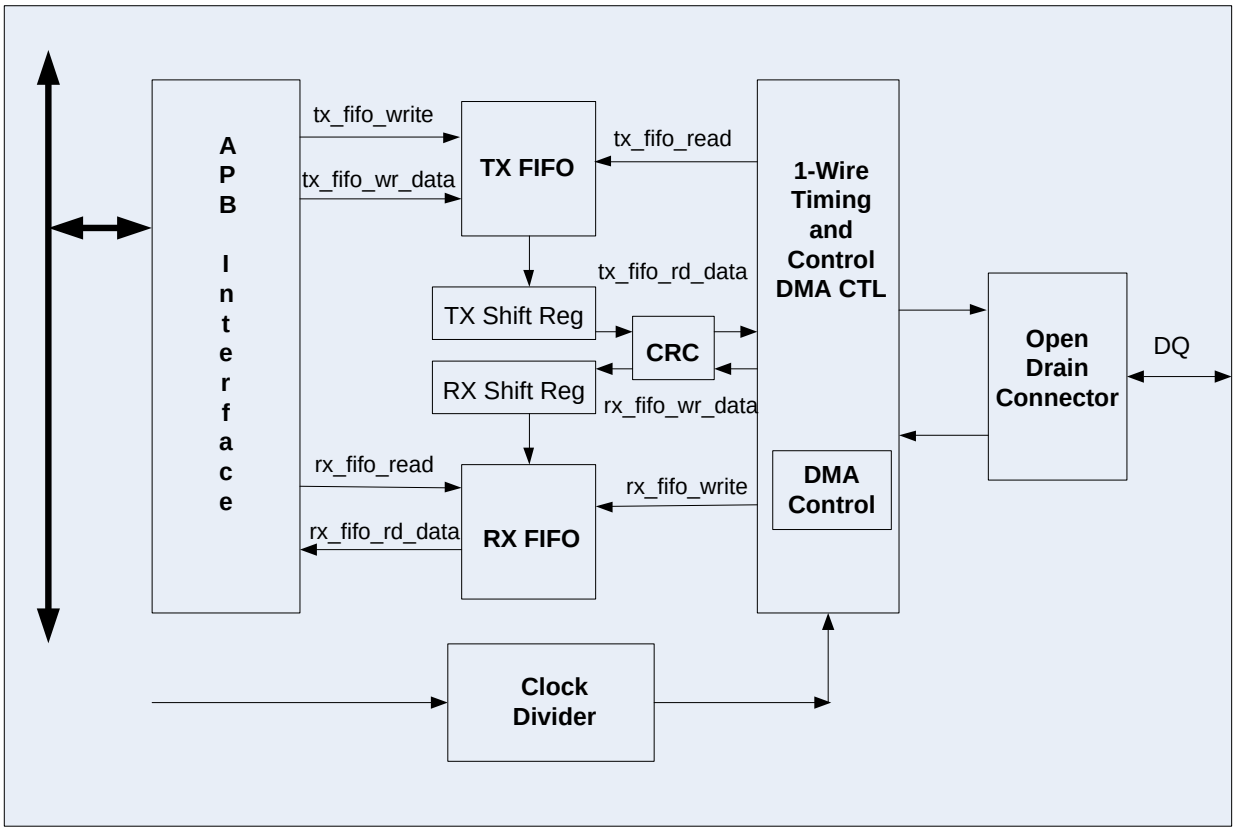
In Tegra® 4 devices, the one-wire protocol is intended to communicate with battery controller chips.

Features

- Independent RX and TX FIFOs
- FIFO depth of 32 x 32 bits
- Hardwired implementation of one-wire protocol to eliminate the need for an external bridge chip
- Software programmable registers
- Software readable status registers
- Receive transmission with interrupt-based operation
- Can run at 1 MHz and higher frequencies; provided timing parameters are programmed
- Deglitch support
- Byte transfers or 1-bit transfers
- Supports the following commands: Read ROM, Skip ROM, Read Memory, Read Status, Read Data/Generate 8-bit CRC, Write Memory, Write Status
- CRC 8-/16-bit implementation support
- Supports different battery devices, up to a memory size of 256K in byte transfer
- Also supports generic devices, such as DS2784. The features supported in DS2784 are:
  - Command
  - Command and Address
  - Command, Address and Write Command
  - Command, Address and Read Command

38.1 Functionality

Figure 157: OWR Block Diagram



Specific command sequences as per the OWR protocol have to be followed for any data transaction over the OWR interface.

The protocol involves first issuing a reset command where the controller looks for a presence from the device, and then issuing ROM commands before the EPROM is accessible, once the presence pulse is received.

The following ROM commands are supported:

- Read ROM (33hex command)
- Skip ROM (CChex command)

When the read ROM command is issued, the controller receives a 64-bit ID from the device. This is an optional command, in case there is only a single slave present. One can directly issue a Skip ROM Command and issue memory commands to access EPROM. When more than one slave is connected, then issue Search ROM and Match ROM commands, to search the particular slave and enable it to interact with the controller implemented using 1-bit mode.

EPROM accessible commands are:

- Read Memory (F0hex command)
- Read Status (AAhex command)
- Read Data/Generate 8 Bit CRC (C3hex command)
- Write Memory (0Fhex command)
- Write Status (55hex command)

Other commands can be used with 1-bit read/write. Contact your NVIDIA FAE for more details.

38.1.1 CRC Generation

To ensure data integrity, the controller has 8-bit and 16-bit CRC calculations. The equivalent polynomial function of this CRC is: $X^8 + X^5 + X^4 + 1$.

The CRC is calculated on the data being received from the device. During a Read Memory command, the software has a provision to enable the CRC check bit where the CRC is calculated on the data received from the device (RD_MEM_CRC_REQ bit in the Control register). When this bit is enabled, after the data is read from the EPROM, the device sends the CRC to the controller .The controller then checks the received CRC against the calculated CRC and sets a crc_error flag in case of mismatch. The CRC can be 8 bits or 16 bits wide depending on the CRC_16BIT_EN bit in the Control register. By programming the BY_PASS_CRC_ERR bit in the Control register, the crc_error is ignored by the controller and continues with the command operation.

For other commands, such as read data CRC, read status, and write memory commands, the CRC checks are done by default as part of the protocol, unlike in read memory where it is optional.

Note: CRC 16-bit is also supported in some battery devices.

38.1.2 OWR Data Transfer Sequence

The OWR protocol consists of five types of data transfer:

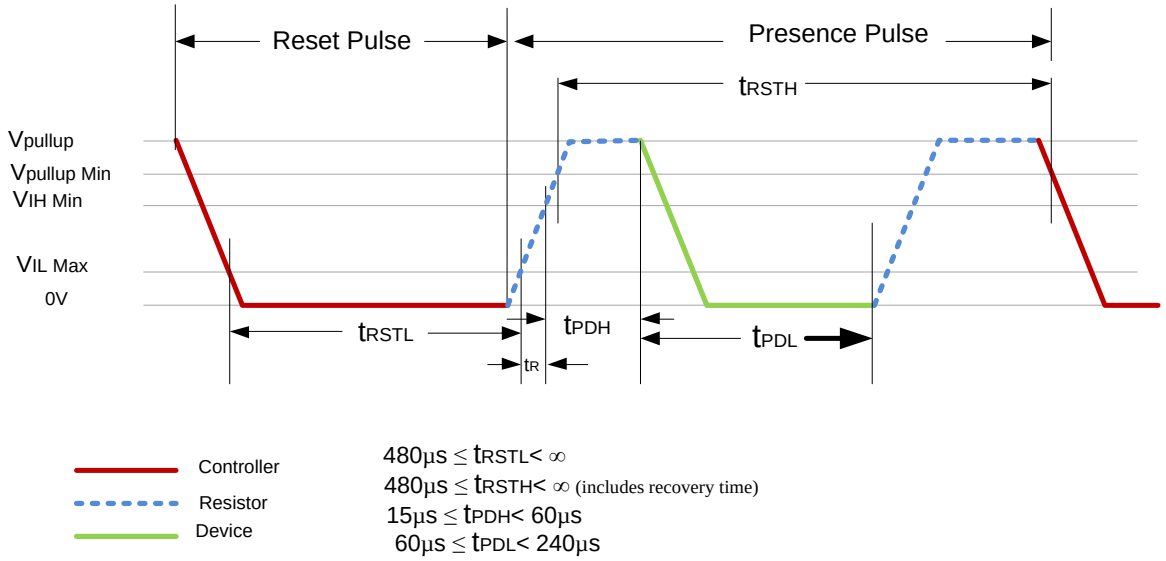
- Reset and presence pulse
- Write 0
- Write 1
- Read Data
- Program Pulse

All data transfers (except presence pulse) are initiated by the controller. The communication should always begin with the initialization sequence, shown in Figure 158 below.

38.1.2.1 Initialization Procedure

To begin any communications on OWR, initialization procedure must be issued. A reset pulse of t_{RSTL} is issued by the controller, and then the controller releases the 1-wire bus, until t_{RSTH} time. After reset pulse, controller detects for rising edge and waits for t_{PDH} time and transmits the presence pulse of t_{PDL}. .

Figure 158: Initialization Sequence



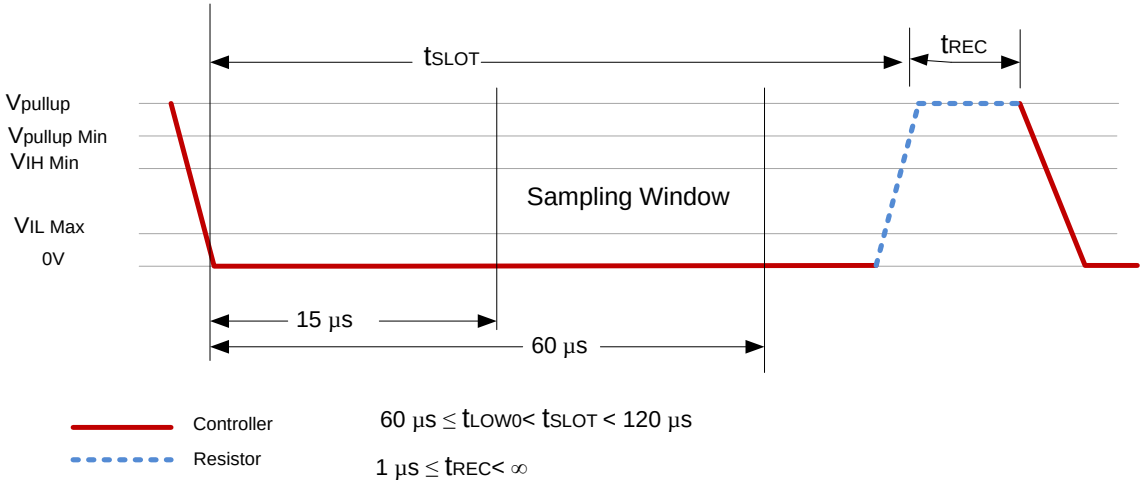
The reset pulse begins the initialization sequence. It is initiated when the GO bit in the Control register (Control.GO) is set.

38.1.2.2 Read/Write Time Slots

Write 0 Timing Slot

During a write memory command, for writing a bit 0, the data can be sampled and written into the scratchpad any time between t_{LOW0} and t_{SLOT} duration as shown in Figure 159 below.

Figure 159: Write 0 Timing



Write 1/Read Data Timing Slot

The Write 1 and Read timing slots are identical. The data bus is pulled low for t_{LOW1} time. The device can sample the data in sampling window (Figure 160). Similarly, during a read slot, the controller can read the data from the device in the sampling window (Figure 161)

Figure 160: Write 1 Timing

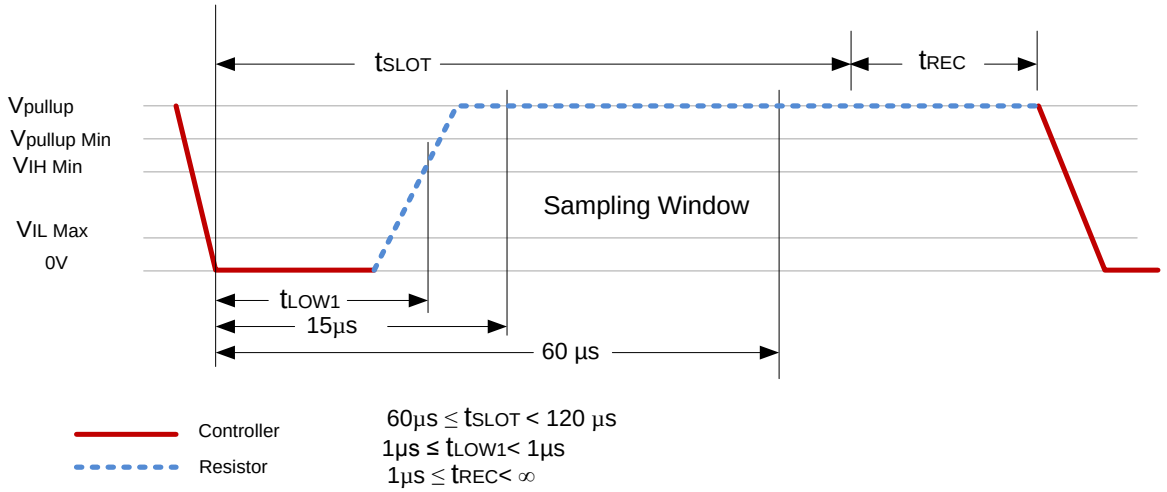
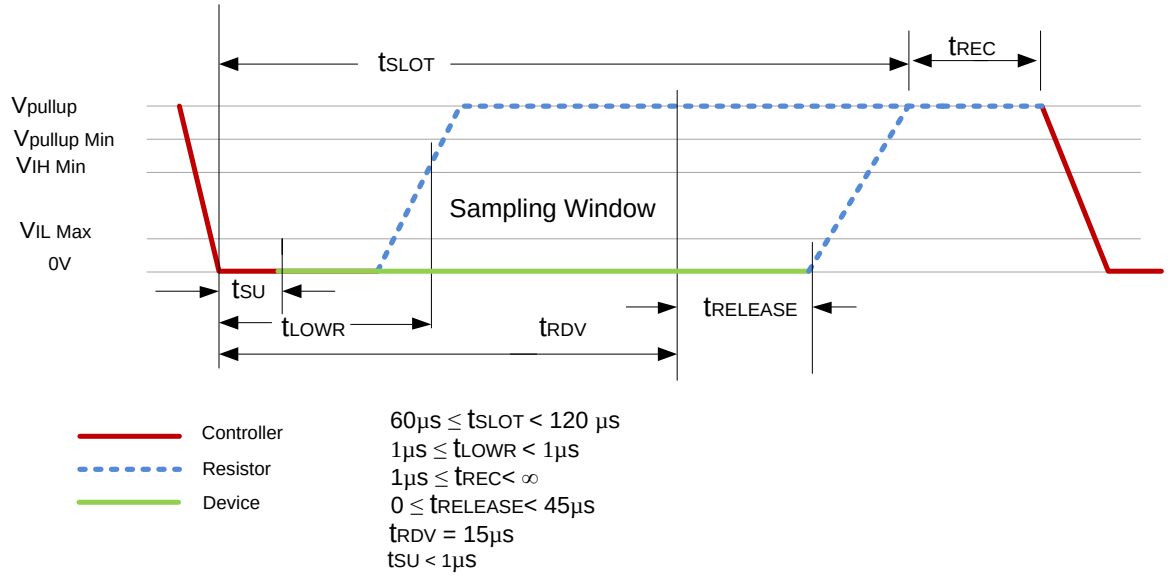


Figure 161: Read Timing

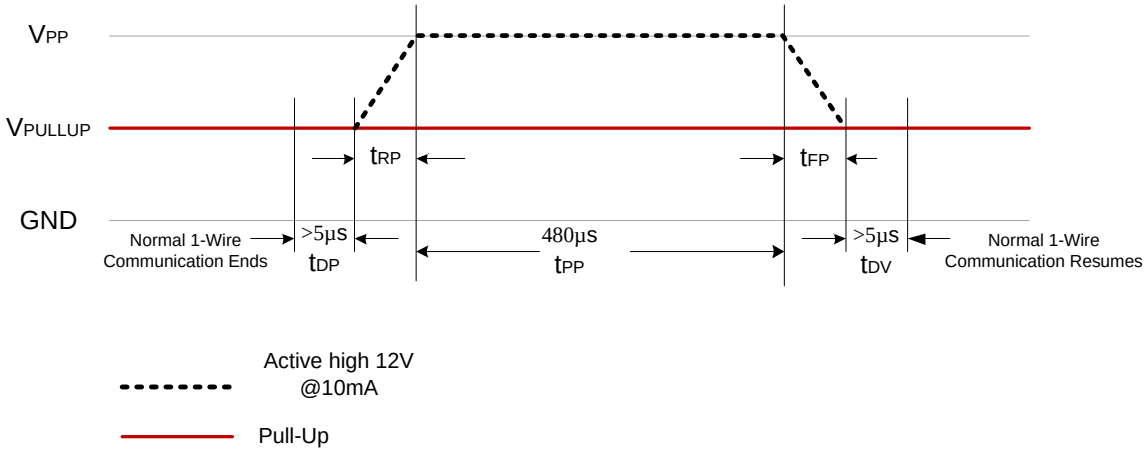


38.1.2.3 Programming Pulse

During a write memory command, data is first copied to the scratchpad. To shift the data from the scratchpad to the EPROM, a program pulse of 12 volts is applied to the data line after the controller has confirmed that the CRC is correct.

- In byte mode transfers, the controller controls the generation of programming pulse.
- In single-bit Write mode, the programming pulse is generated using GPIOs
- In Programming pulse window, the OWR bus is controlled by 12V voltage providing a minimum 10 mA of current.

Figure 162: Programming Timing Pulse

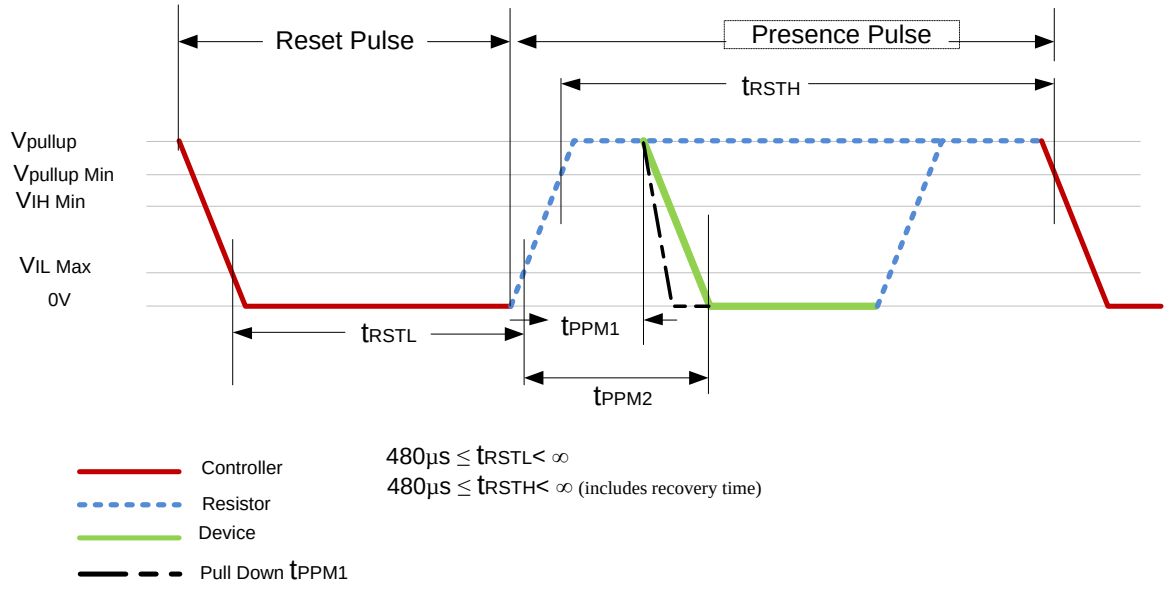


38.1.2.4 Presence-Pulse Masking (PPM)

Presence-Pulse Masking is used when battery devices are propagated through networks and get reflected. Glitches may occur due to reflections in network which may cause slave devices to lose synchronization.

When Presence-Pulse Masking is enabled, the controller pulls the OWR bus to low at t_{PPM1} until t_{PPM2}. At t_{PPM2} battery device further pulls down to low.

Figure 163: Presence-Pulse Masking



38.1.3 Interrupts

Interrupts are used to indicate the status of the design at different stages, so that the next data transfer can be determined.

Done signals indicate the data transfer is done without any interruptions.

Error signals indicate an error has occurred and data transfer should start again. An error can be a CRC error, a presence error, a memory write error, etc.

Overflow/underrun indicates the FIFO status.

Data request indicates FIFOs are ready to accept or send the data.

All the interrupts can be controlled by Interrupt MASK enables and the Interrupt SET register.

38.1.4 Error Handling

If the OWR controller receives an error such as a CRC error, presence error, memory write error, or an error command, a reset pulse must be issued, and the entire sequence must be repeated.

The OWR controller can bypass these errors by enabling register bits in the Control register.

CTL.BYPASS_CRC_ERR: Transfer does not stop on a CRC error

CTL.BYPASS_DGLITCH: Stops the execution of deglitch logic

CTL.PRESENCE_PULSE_MASKING: Transfer does not stop on a presence error.

38.1.5 Clock Control

OWR controller clocking has following clock source options:

Option 0: pllP_out0 = 432 MHz (Fixed 432 MHz directly from PLLP)

Option 1: pllC_out0 = Up to 600 MHz, Programmable clock output directly from PLLC.

Option 2: pllM_out0 = Up to 400 MHz, Programmable clock output directly from PLLM.

Option 3: dbg_oscout = External clock source determined by the customer. 12 MHz, 13 MHz, 19.2 MHz, or 26 MHz are supported frequencies.

An 8-bit divider is used to generate a 1 MHz clock

The clock divider generates a 1 MHz clock that is used as a time reference by the state machine. Transitions between the states of the state machine as well as actions triggered at precise time deadlines are expressed using the 1-MHz clock. The state machine performs all required actions to dialog with the external device.

38.1.6 Deglitch/Debounce

To overcome the glitches on the input DQ line, deglitch logic is used to make sure the data is stable for at least one microsecond. If a glitch occurs, an interrupt is generated and the transfer should start again. There is a bit to bypass deglitch logic. If that bit is set, deglitch is bypassed, and, if glitches occur, they are not reported.

When to sample the data is determined by programming the control register presence_sample_clk for reset initialization and read_data_sample_clk for read time slots.

There is a limitation to the value programmed in the presence/read_data sample clock: the hardware requires 6 clock cycles to implement deglitch generation logic The synchronizer output requires 2 clocks and deglitch requires 1 clock. If there is a glitch on the DQ line, hardware samples the data on the next clock edge, 2 clocks later, to store the sampled data’s 1 clock pulse).

Presence sample clk = Tpdl - 6 clocks

Rd_data_sample_clk = Trdv - 6 clocks

38.1.7 Read/Write 1 Bit

The controller also supports sending or receiving data bit by bit. In a bit-by-bit transfer, the software:

- calculates all the required calculations like when to send the wr0/wr1/rd
- calculates the CRC for received data
- checks CRC and
- preserves the received data

The hardware does not preserve any data in registers, except the received 1 bit data, in the RD_SAMPLE_DATA register.

The controller can only send the Time slots (Wr1/Wr0/Rd) and store the received sampled data in a register. Time slots are controlled by the value programmed in the timing registers.

In case of writes, the program pulse generation can be done using GPIO pins.

38.1.7.1 Programming Sequence

OWR communication always begins with reset initialization. To start reset initialization, program the CTRL.GO bit in the control register. Program all the timing registers before the GO bit. If device presence is detected, program the ROM command and then program the Memory Command.

Programming the ROM Command

Transmit LSB first to send the Read ROM Command (33h, 0011-0011).

Time Slots

- The LSB of the Read ROM command is 1'b1. Program CTRL.WR1_BIT (if 1'b0, program the CTRL.WR0_BIT) in the control register. After the controller generates the write-1 time slot, wait for the write-1 time slot to be completed by looking at INTR_STATUS.BIT_TRANSFER_DONE. Continue this until all the bits are transferred.
- After the Read ROM Command is sent, read 64 bits of ROM data. To receive 64 bits, 64 read time slots have to be sent. To generate the Read time slot, program CTRL.RD_BIT in the control register. After the controller generates the read time slot, the read bit is stored in READ_SAMPLED_BIT. The sampled bit should always be read after the bit transfer is done.
- Next, program the CTRL.RD_BIT, to read the second bit. The read time slot is generated by the controller. Wait for done and read the sampled bit.

Repeat the sequence for the remaining bits.

After all 64 bits are received, Memory data has to be read.

Reading EPROM Data using the Read Memory Command

- To read Memory data, first send the memory command followed by memory address. After the memory command and address are sent, verify the data sent. This can be done by checking the CRC of the command and address sent. To receive CRC, send 8 read slots, read the CRC data, and check for a match with the computed value. Read the EPROM data from memory or else start with initialization procedure.
- To read data from EPROM, send read time slots. The number of read time slots should be memory address – EPROM end offset. After reading EPROM data, read the CRC and check the received data, if CRC is required. Normally, the CRC is embedded within the EPROM data. To receive 8-bit CRC data, send 8 read time slots.

38.2 Programming Guidelines

38.2.1 Programming Registers

All the registers should be programmed and the CTL.GO bit should be programmed to start the controller.

Registers to Program Prior to the Control Register

- **All timing registers**: Program the timing values according to the battery device data sheet
- **Command register**: Program the device ROM Command, Memory Command, EPROM starting address
- **EPROM size offset**: Program the number of bytes to transfer
- **Interrupt mask enables**: Program the respective interrupt mask enables to enable the interrupts to CPU/COP
- **Control Register**: Programming this register depends on the type of execution required.

Program the GO bit only after programming all of the above.

Registers to Monitor

- **Interrupt Status & Source**: These are used to monitor the design status (like done, error, etc.).
- **CRC**: Stores the calculated CRC 8/16-bit value
- **Byte Count**: Indicates the number of bytes transferred/received
- **Read_ROM0/ROM1**: Stores the received ROM Family code, Serial Number, CRC.

FIFO Registers

- **Tx/Rx FIFO**: Data to transmit/receive data.

Notes:

- The Rd_data_crc bit needs to be enabled with both Read and Write commands in case of generic CRC Read and Write
- For Write commands, the CmdAddr CRC bit should be disabled
- Read memory command for both DS2502 and DS2784 can be supported in generic mode.
- DS2502 requires a programming pulse which is not supported by generic mode.

38.2.2 Byte Data Transfer Sequence

Reset Initialization → ROM Command → Memory Command → Reset Initialization → ROM Command → Memory Command, etc.

In case of an error, the entire sequence should start again from the beginning:

Reset Initialization → Error → Reset Initialization →ROM Command → Error → Reset Initialization → ROM Command → Memory Command, etc.

38.3 OWR Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

The 1-wire interface provides the communication line to a 1-Kbit Add-Only Memory (DS2502). The required protocol for accessing the DS2502 is defined by Dallas Semiconductor.

The DS2502 holds battery characteristics information. The interface sends or receives one bit or byte at a time.

- Supports 2 ROM commands: Skip ROM, Read ROM.
- Supports 5 Memory commands: Write Memory, Write Status Byte, Read Status Byte, Read Memory, and Read Data/Generate 8-bit CRC
- Different battery devices are also supported, with memory size 256k
- Battery devices with EPROM only are supported, devices with NV_RAMS are not supported

Programming Guidelines

The Controller supports two modes:

- Byte transfer mode
- Bit transfer mode

If the interface sends or receives a byte at a time: All the Timing registers, the command register, and the required fields in control registers should be programmed and then the GO bit in the control register should be programmed. ROM CMD data is stored in the ROM registers. EPROM data/status is stored in FIFOs. Interrupt bits know the status of the design and can be enabled or disabled.

If the interface sends or receives one bit at a time: This mode is enabled by enabling DATA_TRANSFER_MODE in the control register. All the timing registers should be programmed. Software writes WR0_BIT, WR1_BIT, or RD_BIT to enable the write one time slot, write zero time slot, or read time slot, wr0_busy, wr1_busy, and rd_busy to indicate the data transfer of the bit.

To read the sampled data: the sampled data is valid only after rd_busy goes low and an interrupt bit to indicate bit transfer is ready to accept the next bit. In one bit transfer mode, CRC calculations to detect errors and when to send write slots and read slots is decided by software only. One bit transfer mode supports only read commands.

38.3.1 OWR_CONTROL_0

OWR Control Register

This register is the main control register. It should be configured last after all other settings are configured.

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	RD_BIT: If 0, no transfer is done. If 1, the read time slot is executed. Reads of this write-only bit will return 0. 0 = NO_TRANSFER 1 = TRANSFER_READ_SLOT
30	0x0	WR0_BIT: If 0, no transfer is done. If 1, write zero time slot is executed. Reads of this write-only bit will return 0. 0 = NO_TRANSFER 1 = TRANSFER_ZERO
29	0x0	WR1_BIT: If 0, no transfer is done. If 1, write one time slot is executed. Reads of this write-only bit will return 0. 0 = NO_TRANSFER 1 = TRANSFER_ONE
28	0x0	BY_PASS_CRC_ERR: This bit is set to 1, if the transfer needs to continue on CRC error. Otherwise, on error the transfer stops, and the transfer should start on setting RPP reset (GO bit). 0 = STOP_TRANSFER_ON_CRC_ERR 1 = CONTINUE_TRANSFER_ON_CRC_ERR

Bit	Reset	Description
27	0x0	BY_PASS_DGLITCH: This bit is used to bypass the deglitch logic. If 1, just the sync output is taken. If 0, looks for any glitch in the sample window for at least 1 µs, Deglitch requires a minimum of 6 clocks (2 for sync, 2 for deglitch, and if glitch, checks for 2 more clocks). If glitch still exists, an error interrupt is asserted and the data transfer should start from first. 0 = START_DGLITCH 1 = NO_DGLITCH
26:23	0x0	RD_DATA_SAMPLE_CLK: Read the data sample window. The master samples data_in, which should be less than or equal to (tlow1 - 6) clocks. 6 clocks are used for deglitch, if deglitch is bypassed, 3 clocks should be enough.
22:15	0x0	PRESENCE_SAMPLE_CLK: Presence pulse sample clock. The master samples data_in, which should be less than or equal to (tpdl - 6) clocks. 6 clocks are used for deglitch, if deglitch is bypassed, 3 clocks should be enough.
14	0x0	RD_MEM_CRC_REQ: This bit is set to 1, if CRC is required for read memory command at the end of memory. 0 = NO_CRC_READ 1 = CRC_READ
13:9	0x0	RX_FIFO_ATTEN_LEVEL: Receive FIFO attention level. 000 = 1 word, FIFO request is asserted when at least one word is full in the FIFO. 001 = 2 word, FIFO request is asserted when at least 2 words are full in the FIFO, etc.
8:4	0x0	TX_FIFO_ATTEN_LEVEL: Transmit FIFO attention level. 000 = 1 word, FIFO request is asserted when at least one word is empty in the FIFO. 001 = 2 word, FIFO request is asserted when at least 2 words are empty in the FIFO, etc.
3	0x0	CRC_16BIT_EN: If set to 1, 16-bit CRC is executed. If set to 0, 8-bit CRC is executed. 0 = CRC_8BIT_EN 1 = CRC_16BIT_EN
2	0x0	DATA_TRANSFER_MODE: If set to 1, data transfer is done bit by bit. If set to 0, data transfer is done in bytes. 0 = BYTE_TRANSFER_MODE 1 = BIT_TRANSFER_MODE
1	0x0	PRESENCE_PULSE_MASKING: When set, DQ is driven to low by the master before the slave does. Clearing this bit disables the PPM. 0 = NO_PPM 1 = START_PPM
0	0x0	GO: Generate Reset Presence Pulse write only bit. Reads to this register will return 0. This bit should be programmed after all the registers are programmed. 0 = NO_PRESENCE_PULSE 1 = START_PRESENCE_PULSE

38.3.2 OWR_COMMAND_0

OWR Command Register

ROM_CMD, MEM_CMD, and MEM_ADDR should be programmed at the same time.

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

Bit	Reset	Description
31:16	0x0	MEM_ADDR: EPROM Starting Address[15:0] to write/read data into EPROM
15:8	0x0	MEM_CMD: 1-wire MEM commands
7:0	0x0	ROM_CMD: 1-wire ROM commands

38.3.3 OWR_EPROM_0

OWR EPROM Size Offset

This register indicates the number of bytes to transfer. Up to 256k EPROM size is supported.

MEM_ADDR and MEMORY_BYTES_TRANSFER/STATUS_BYTES_TRANSFER should always be in sync. For example, if the EPROM end offset address is 0x7f, and we want to read data from the fourth location address of EPROM, the programming should be as follows:

- MEM_ADDR = 0x4;
- MEMORY_BYTES_TRANSFER = 0x7B; (0x4 - 0x7f)

The steps are similar for status bytes.

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

Bit	Reset	Description
31:16	0x0	STATUS_BYTES_TRANSFER: Number of EPROM Status bytes to transfer: Mem_Addr - Status bytes end address
15:0	0x0	MEMORY_BYTES_TRANSFER: Number of EPROM memory bytes to transfer: Mem_Addr - EPROM end address

38.3.4 OWR_WR_RD_TCTL_0

OWR Write Read Timing Control Register

Controls the one-wire DQ output (Pullup or Pulldown) to write and read time slots. Timings are in microseconds.

Note : <= means less than or equal to.

Offset: 0xc | Read/Write: R/W | Reset: 0x00000000 (0bxx000000000000000000000000000000)

Bit	Reset	Description
29:28	0x0	TSU: Read Data Setup, Tsu = N OWR clocks, Range = tsu < 1
27:22	0x0	TRELEASE: Release 1-wire Time, Trelease = N OWR clocks, Range = 0 <= trelease < 45
21:18	0x0	TRDV: Read data valid time, Trdv = N+1 OWR clocks, Range = Exactly 15
17:11	0x0	TLOW0: Write Zero time Low, Tlow0 = N+1 OWR clocks, Range = 60 <= tlow0 < tslot < 120
10:7	0x0	TLOW1: Write one time Low, or TLOWR both are same. Tlow1 = N+1 OWR clocks, Range = 1 <= tlow1 < 15. TlowR = N+1 OWR clocks, Range = 1 <= tlowR < 15
6:0	0x0	TSLOT: Active time slot for write or read data, Tslot = N+1 OWR clocks, Range = 60 <= tslot < 120

38.3.5 OWR_RST_PRESENCE_TCTL_0

OWR Reset Presence Timing Control Register

Controls the one-wire DQ output (Pullup or Pulldown) to reset initialization time slots. Timings are in microseconds.

Note : <= means less than or equal to.

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	TPDL: PRESENCE_DETECT_LOW Tpdl = N OWR clocks, Range = 60 <= tpdl < 240
23:18	0x0	TPDH: PRESENCE_DETECT_HIGH Tpdh = N+1 OWR clocks, Range = 15 <= tpdh < 60

Bit	Reset	Description
17:9	0x0	TRSTL: RESET_TIME_LOW Trstl = N+1 OWR clocks, Range = 480 <= trstl < infinity
8:0	0x0	TRSTH: RESET_TIME_HIGH, Trsth = N+1 OWR clocks, Range = 480 <= trsth < infinity

38.3.6 OWR_PPM_CORRECTION_TCTL_0

OWR PRESENCE PULSE MASKING TIMING CONTROL

Drives the DQ line to low before the slave does.

Offset: 0x14 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:6	0x0	TPPM2: PRESENCE PULSE MASK STOP
5:0	0x0	TPPM1: PRESENCE PULSE MASK START

38.3.7 OWR_PROG_PULSE_TCTL_0

OWR PROGRAM PULSE TIMING CONTROL Register

Controls the 12V programming pulse used to write data to EPROM. Timings are in microseconds.

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	TPP: Program Pulse Width. Tpp = N OWR clocks. Range = 480 to 5000
15:12	0x0	TFP: Program Voltage Fall Time. Tfp = N OWR clocks. Range = 0.5 to 5
11:8	0x0	TRP: Program Voltage Rise Time. Trp = N OWR clocks. Range = 0.5 to 5
7:4	0x0	TDV: Delay to verify. Tdv = N OWR clocks. Range = > 5
3:0	0x0	TPD: Delay to program. Tpd = N+1 OWR clocks. Range = > 5

38.3.8 OWR_READ_ROM0_0

The Read ROM Command reads the 8-bit family code, 48-bit serial number, and 8-bit CRC for a total of 64 bits. Read ROM Command data is split into two 32-bit registers: READ_ROM0 and READ_ROM1.

OWR READ ROM DATA0 Register

Offset: 0x1c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:8	X	SERIAL_NUM0: Reads the first 24 bits of the ROM serial number
7:0	X	FAMILY_CODE: Reads the 8-bit family code of the ROM

38.3.9 OWR_READ_ROM1_0

OWR READ ROM DATA1 Register

Offset: 0x20 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:24	X	CRC_BYTE: Reads the 8-bit CRC code of the ROM
23:0	X	SERIAL_NUM1: Reads the next 24 bits of the ROM serial number

38.3.10 OWR_INTR_MASK_0

OWR INTR Mask Register

This register stores the masks for the interrupts. If a bit is set to 1 and the corresponding interrupt condition is satisfied, the interrupt line to the system interrupt controller is asserted.

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxx00000000000000)

Bit	Reset	Description
13	0x0	BIT_TRANSFER_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
12	0x0	RXFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
11	0x0	TXFIFO_DATA_REQ_INT_EN: 0 = DISABLE 1 = ENABLE
10	0x0	DGLITCH_INT_EN: 0 = DISABLE 1 = ENABLE
9	0x0	RXF_UNR_INT_EN: 0 = DISABLE 1 = ENABLE
8	0x0	TXF_OVF_INT_EN: 0 = DISABLE 1 = ENABLE
7	0x0	MEM_CMD_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
6	0x0	ROM_CMD_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
5	0x0	PRESENCE_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
4	0x0	RESET_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
3	0x0	ERR_CMD_INT_EN: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	MEM_WR_ERR_INT_EN: 0 = DISABLE 1 = ENABLE
1	0x0	CRC_ERR_INT_EN: 0 = DISABLE 1 = ENABLE
0	0x0	PRESENCE_ERR_INT_EN: 0 = DISABLE 1 = ENABLE

38.3.11 OWR_INTR_STATUS_0

OWR Status Register

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000XX00 (0bxxxxxxxxxxxxxxxxxxx0xx00000000000)

Bit	R/W	Reset	Description
13	RW	0x0	BIT_TRANSFER_DONE: This bit is set when the transfer of each bit is done. This is set in one bit transfer mode. Software writes a 1 to clear this bit. 0 = NOT_DONE 1 = DONE
12	RO	X	RXFIFO_DATA_REQ: RX FIFO data request 0 = RX_NOT_RDY 1 = RX_RDY
11	RO	X	TXFIFO_DATA_REQ: TX FIFO data request 0 = TX_NOT_RDY 1 = TX_RDY
10	RW	0x0	DGLITCH: This bit is set when data is not stable for at least 1 µs. Software writes a 1 to clear this bit. If deglitch is detected, the data transfer should start from first. 0 = DGLITCH_NOT_DETECTED 1 = DGLITCH_DETECTED
9	RW	0x0	RXF_UNR: RX FIFO Under run. This bit is set to 1 whenever software tries to read from an empty RX FIFO. Software writes a 1 to clear this bit. 0 = NOT_EMPTY 1 = EMPTY
8	RW	0x0	TXF_OVF: TX FIFO Overflow. This bit is set to 1 whenever software tries to write to a full TX FIFO. Software writes a 1 to clear this bit. 0 = NOT_EMPTY 1 = EMPTY
7	RW	0x0	MEM_CMD_DONE: This indicates the master has written data into EPROM or data was received from EPROM without any error. Software writes a 1 to clear it. 0 = NOT_DONE 1 = DONE
6	RW	0x0	ROM_CMD_DONE: This indicates the master has received the ROM data from the battery. Software writes a 1 to clear it. 0 = NOT_DONE 1 = DONE
5	RW	0x0	PRESENCE_DONE: This indicates the presence is done, and the master has detected the device. Software writes a 1 to clear it. 0 = NOT_DONE 1 = DONE
4	RW	0x0	RESET_DONE: This indicates the master has sent the reset, then waits for presence. Software writes a 1 to clear it. 0 = NOT_DONE 1 = DONE

Bit	R/W	Reset	Description
3	RW	0x0	ERR_CMD: ERROR CMD: Indicates an error command was written in the register. It should be ignored when the transfer is in single bit mode. Software writes a 1 to clear it. 0 = CORRECT_CMD 1 = ERROR_CMD
2	RW	0x0	MEM_WR_ERR: MEM WR Error: Indicates whether the received data from EPROM is correct or not. Software writes a 1 to clear it. 0 = N0_ERROR 1 = ERROR
1	RW	0x0	CRC_ERR: CRC Error: Indicates whether the received data is correct or not. Software writes a 1 to clear it. 0 = N0_ERROR 1 = ERROR
0	RW	0x0	PRESENCE_ERR: Presence Error. This bit is set when device presence is not found. 0 = SLAVE_DETECTED 1 = NO_SLAVE_DETECTED

38.3.12 OWR_INTR_SOURCE_0

OWR INTR Source Register

This read-only register returns the AND of the Interrupt Source and Interrupt Mask registers.

Offset: 0x2c | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
13	X	BIT_TRANSFER_DONE: 0 = NOT_DONE 1 = DONE
12	X	RXFIFO_DATA_REQ: 0 = RX_NOT_RDY 1 = RX_RDY
11	X	TXFIFO_DATA_REQ: 0 = TX_NOT_RDY 1 = TX_RDY
10	X	DGLITCH: 0 = DGLITCH_NOT_DETECTED 1 = DGLITCH_DETECTED
9	X	RXF_UNR: 0 = NOT_EMPTY 1 = EMPTY
8	X	TXF_OVF: 0 = NOT_EMPTY 1 = EMPTY
7	X	MEM_CMD_DONE: 0 = NOT_DONE 1 = DONE
6	X	ROM_CMD_DONE: 0 = NOT_DONE 1 = DONE
5	X	PRESENCE_DONE: 0 = NOT_DONE 1 = DONE
4	X	RESET_DONE: 0 = NOT_DONE 1 = DONE

Bit	Reset	Description
3	X	ERR_CMD: 0 = CORRECT_CMD 1 = ERROR_CMD
2	X	MEM_WR_ERR: 0 = N0_ERROR 1 = ERROR
1	X	CRC_ERR: 0 = N0_ERROR 1 = ERROR
0	X	PRESENCE_ERR: 0 = SLAVE_DETECTED 1 = NO_SLAVE_DETECTED

38.3.13 OWR_INTR_SET_0

Interrupt Set Register

This write-only register can be used to set the interrupt status bit. A write to this register causes the bits in the status register to be set to 1. Reads always return 0.

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxx0xx00000000000)

Bit	Reset	Description
13	0x0	BIT_TRANSFER_DONE: 0 = NOT_DONE 1 = DONE
10	0x0	DGLITCH: 0 = DGLITCH_NOT_DETECTED 1 = DGLITCH_DETECTED
9	0x0	RXF_UNR: 0 = NOT_EMPTY 1 = EMPTY
8	0x0	TXF_OVF: 0 = NOT_EMPTY 1 = EMPTY
7	0x0	MEM_CMD_DONE: 0 = NOT_DONE 1 = DONE
6	0x0	ROM_CMD_DONE: 0 = NOT_DONE 1 = DONE
5	0x0	PRESENCE_DONE: 0 = NOT_DONE 1 = DONE
4	0x0	RESET_DONE: 0 = NOT_DONE 1 = DONE
3	0x0	ERR_CMD: 0 = CORRECT_CMD 1 = ERROR_CMD
2	0x0	MEM_WR_ERR: 0 = N0_ERROR 1 = ERROR

Bit	Reset	Description
1	0x0	CRC_ERR: 0 = N0_ERROR 1 = ERROR
0	0x0	PRESENCE_ERR: 0 = SLAVE_DETECTED 1 = NO_SLAVE_DETECTED

38.3.14 OWR_STATUS_0

Offset: 0x34 | Read/Write: R/W | Reset: 0x00XXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00xxxxx)

Bit	R/W	Reset	Description
23	RO	X	READ_SAMPLED_BIT: The bit is valid only when RD_BUSY is cleared. 0 = READ_ZERO 1 = READ_ONE
22	RO	X	RD_BUSY: READ : This self-clearing bit is cleared when a read time slot completes on the read sequence, and the sampled read bit is stored in READ_BIT 0 = IDLE 1 = BUSY
21	RO	X	WR1_BUSY: WRITE1: This self-clearing bit is cleared when a write one time slot completes on write sequence 1. 0 = IDLE 1 = BUSY
20	RO	X	WR0_BUSY: WRITE 0: This self-clearing bit is cleared when a write zero time slot completes on write sequence 0. 0 = IDLE 1 = BUSY
19	RO	X	RPP: This is set when the RPP reset bit is set in the control register (GO), auto-cleared on completion of the reset initialization sequence. 0 = IDLE 1 = RESET_PRESENCE_PULSE
18:13	RO	X	TX_FIFO_EMPTY_CNT: The number of slots that can be written to the TX FIFO.
12:7	RO	X	RX_FIFO_FULL_CNT: The number of slots to be read from the RX FIFO.
6	RW	0x0	RX_FLUSH: Flushes the RX FIFO, cleared after the FIFO is empty. 0 = DISABLE 1 = ENABLE
5	RW	0x0	TX_FLUSH: Flushes the TX FIFO, cleared after the FIFO is empty. 0 = DISABLE 1 = ENABLE
4	RO	X	RXF_EMPTY: RX FIFO empty status: RO. Hardware sets this bit to 1, if the RX FIFO is empty. Otherwise, this bit is set to 0. 0 = NOT_EMPTY 1 = EMPTY
3	RO	X	RXF_FULL: RX FIFO full status: Hardware sets this bit to 1, if the RX FIFO is full. Otherwise, this bit is set to 0. 0 = NOT_FULL 1 = FULL
2	RO	X	TXF_EMPTY: TX FIFO empty status: Hardware sets this bit to 1, if the TX FIFO is empty. Otherwise, this bit is set to 0. 0 = NOT_EMPTY 1 = EMPTY
1	RO	X	TXF_FULL: TX FIFO full status: Hardware sets this bit to 1 if the TX FIFO is full. Otherwise, this bit is set to 0. 0 = NOT_FULL 1 = FULL

Bit	R/W	Reset	Description
0	RO	X	RDY: Ready bit. This bit is set at the end of every transfer, and it is cleared by hardware when the next transfer starts. 0 = NOT_READY 1 = READY

38.3.15 OWR_CRC_0

OWR CRC Register

Offset: 0x38 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	CRC_CALC: CRC calculated by OWR current Write/Read operation
15:0	X	CRC_RECEV: CRC received on read data

38.3.16 OWR_BYTE_CNT_0

OWR BYTE_CNT Register

Offset: 0x3c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TRANSMITTED: Number of bytes transmitted on write commands or address sent
15:0	X	RECEIVED: Number of bytes received on read data includes CRC byte count

38.3.17 OWR_TX_FIFO_0

OWR TX FIFO Register

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	WR_DATA: TX FIFO

38.3.18 OWR_RX_FIFO_0

OWR RX FIFO Register

Offset: 0x44 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	RD_DATA: RX FIFO

38.3.19 OWR_RECOVERY_TIME_TCTL_0

OWR RECOVERY TIME Register

Offset: 0x4c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TREC: Recovery time slot for write or read data. Trec= N+1 OWR clocks

38.3.20 OWR_RST_TIME_TCTL_0

OWR RST PRESENCE HIGH AND LOW TIME Register

Offset: 0x50 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	TRSTL: RESET_TIME_LOW Trstl = N+1 OWR clocks
15:0	0x0	TRSTH: RESET_TIME_HIGH, Trsth = N+1 OWR clocks

38.3.21 OWR_RST_PRESENCE_DETECT_TCTL_0

OWR RST PRESENCE DETECT HIGH AND LOW TIME Register

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	TPDL: PRESENCE_DETECT_LOW Tpdl = N OWR clocks
15:0	0x0	TPDH: PRESENCE_DETECT_HIGH Tpdh = N+1 OWR clocks

38.3.22 OWR_WRITE_ONE_ZERO_TCTL_0

OWR WRITE 1 OR WRITE 0 TIME Register

Offset: 0x58 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	TLOW0: Write zero time Low. Tlow0 = N+1 OWR clocks
15:0	0x0	TLOW1: Write one time Low, or TLOWR both are same. Tlow1 = N+1 OWR clocks. TlowR = N+1 OWR clocks

38.3.23 OWR_TIME_SLOT_RELEASE_TCTL_0

OWR TIME SLOT OR RELEASE TIME Register

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	TRELEASE: Release 1-wire time. Trelease = N OWR clocks
15:0	0x0	TSLOT: Active time slot for write or read data. Tslot = N+1 OWR clocks

38.3.24 OWR_READ_DATA_TCTL_0

OWR WRITE 1 OR WRITE 0 LOW TIME Register

Offset: 0x60 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	TRDV: Read data valid time. Trdv = N+1 OWR clocks
15:0	0x0	TSU: Read Data Setup. Tsu = N OWR clocks

38.3.25 OWR_PROGRAM_PULSE_TCTL_0

OWR PROGRAM PULSE TIMING CONTROL Register

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:24	0x0	TFP: Program Voltage Fall Time. Tfp = N OWR clocks
23:16	0x0	TRP: Program Voltage Rise Time. Trp = N OWR clocks
15:8	0x0	TDV: Delay to verify. Tdv = N OWR clocks
7:0	0x0	TPD: Delay to program. Tpd = N+1 OWR clocks

38.3.26 OWR_PROGRAM_PULSE_WIDTH_TCTL_0

OWR PROGRAM PULSE TIMING CONTROL Register

Offset: 0x68 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:0	0x0	TPP: Program Pulse Width. Tpp = N OWR clocks. Range = 480 to 5000.

38.3.27 OWR_PPM_TCTL_0

OWR PRESENCE PULSE MASKING TIMING CONTROL

Offset: 0x6c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	TPPM2: PRESENCE PULSE MASK STOP
15:0	0x0	TPPM1: PRESENCE PULSE MASK START

38.3.28 OWR_SAMPLE_DATA_0

OWR SAMPLE DATA Register

Offset: 0x70 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	RD_DATA_SAMPLE_CLK_EN: 0 = DISABLE 1 = ENABLE
30:16	0x0	RD_DATA_SAMPLE_CLK: Read data sample window. The master samples data_in, which should be less than or equal to (tlow1 - 6) clocks. 6 clocks are used for deglitch. If deglitch is bypassed, 3 clocks should be enough
15	0x0	PRESENCE_SAMPLE_CLK_EN: 0 = DISABLE 1 = ENABLE
14:0	0x0	PRESENCE_SAMPLE_CLK: Presence pulse sample clock. The master samples data_in, which should be less than or equal to (tpdl - 6) clocks. 6 clocks are used for deglitch. If deglitch is bypassed, 3 clocks should be enough.

38.3.29 OWR_GENERIC_CTL_0

GENERIC OWR COMMAND TRANSFER CONTROL

Offset: 0x74 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx00000000000)

Bit	Reset	Description
10	0x0	RD_DATA_CRC: 0 = NO_READ 1 = READ
9	0x0	CMD_ADDR_CRC: 0 = NO_READ 1 = READ
8	0x0	DATA_XFER_TWAIT: 0 = NO_WAIT 1 = WAIT
7	0x0	WR_DATA_XFER: 0 = NO_WRITE 1 = WRITE
6	0x0	RD_DATA_XFER: 0 = NO_READ 1 = READ
5	0x0	ADDR_XFER_TWAIT: 0 = NO_WAIT 1 = WAIT
4	0x0	ADDR_XFER_MODE: 0 = EN_8BIT 1 = EN_16BIT
3	0x0	ADDR_XFER: 0 = NO_SEND 1 = SEND
2	0x0	CMD_XFER_TWAIT: 0 = NO_WAIT 1 = WAIT
1	0x0	CMD_XFER_WDATA: 0 = NO_WRITE 1 = WRITE
0	0x0	CMD_XFER: 0 = NO_SEND 1 = SEND

38.3.30 OWR_CMD_XFER_TCTL_0

OWR COMMAND TRANSFER TIMING CONTROL Register

Offset: 0x78 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TWAIT

38.3.31 OWR_ADDR_XFER_TCTL_0

OWR ADDRESS TRANSFER TIMING CONTROL Register

Offset: 0x7c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TWAIT

38.3.32 OWR_DATA_XFER_TCTL_0

OWR DATA TRANSFER TIMING CONTROL Register

Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	TWAIT

[THIS PAGE INTENTIONALLY LEFT BLANK]

39.0 PWM CONTROLLER

The Pulse Width Modulator (PWM) controller is a four channel frequency divider whose pulse width varies. Each channel has a programmable frequency divider and a programmable pulse width generator.

Frequency division is a 13-bit programmable value, and pulse division is an 8-bit value.

The PWM runs off a device clock, which is programmed in the Clock and Reset controller, and can be any frequency up to the device clock maximum speed of 48 MHz.

The device clock frequency is always subject to a minimum divide by 256 to generate the PWM output frequency, and may also be subdivided to slow it down further based on the programmable value locally.

There is an APB interface that interfaces the register logic to the APB bus.

The PWM contains four independently programmable pulse width modulators. Each generated pulse has an n/256 duty cycle. PWM signals are useful for LCD contrast and brightness control, VCO-generated clocks and other analog voltage references where high precision is not required.

39.1 Functionality

There are four PWM controllers on four individual pins on the chip. The pinmux corresponding to these are SDC primary pin groups. For the four PWM controllers there are four corresponding registers:

- PWM_CSR0
- PWM_CSR1
- PWM_CSR2
- PWM_CSR3

Each PWM must be enabled (bit [31]) to be operational. Pulse Width [23:16] determines the output pulse width and must be programmed appropriately. Pulse Width [30:24] is not used and must be 0. Frequency Divider [12:0] determines the Divided clock.

39.2 PWM Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

39.2.1 PWM_CONTROLLER_PWM_CSR_0_0

PWM Output-0 Configuration Control Register

Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000xxx0000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable Pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: pulse width that needs to be programmed. 0=Always low. 1=1/256 Pulse high. 2=2/256 Pulse high. N=N/256 Pulse high
12:0	0x0	PFM_0: Frequency divider that needs to be programmed.

39.2.2 PWM_CONTROLLER_PWM_CSR_1_0

PWM Output-1 Configuration Control Register

Offset: 0x10 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000xxx0000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: pulse width that needs to be programmed. 0=Always low. 1=1/256 Pulse high. 2=2/256 Pulse high. N=N/256 Pulse high
12:0	0x0	PFM_1: Frequency divider that needs to be programmed.

39.2.3 PWM_CONTROLLER_PWM_CSR_2_0

PWM Output-2 Configuration Control Register

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000xxx0000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: Pulse Width that needs to be programmed. 0=Always low. 1=1/256 Pulse high. 2=2/256 Pulse high. N=N/256 Pulse high
12:0	0x0	PFM_2: Frequency divider that needs to be programmed.

39.2.4 PWM_CONTROLLER_PWM_CSR_3_0

PWM Output-3 Configuration Control Register

Offset: 0x30 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000xxx0000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable pulse width modulator 0 = DISABLE 1 = ENABLE
30:16	0x0	PWM_0: pulse width that needs to be programmed 0=Always low 1=1/256 Pulse high 2=2/256 Pulse high N=N/256 Pulse high
12:0	0x0	PFM_3: Frequency divider that needs to be programmed.

40.0 THERMAL SENSOR AND THERMAL THROTTLING CONTROLLER

40.1 Thermal Throttling Controller (SOC_THERM)

The goals of the thermal throttling controller include:

- Thermal sensing: Access, capture and processing data from thermal sensors which are located in multiple locations around the SOC.
- Thermal throttling: Providing a means of reducing the chip performance to manage power consumption in response to thermal events. Provide support for several possible software and hardware throttling mechanisms.
- External OC Alarm Slowdown: Providing a means of reducing chip performance to manage power consumption on detecting over-current alarms from outside the Tegra® 4 SOC. Examples of such OC alarms include “AC power unplug” event, PMIC over-current, battery over-current, radio power amplifier over-current events. Some events require the system to be throttled to lower power/performance within latencies that cannot be handled by software alone.
- Throttling in a di/dt safe manner: Providing hardware support to ensure that throttling in response to over-temp and over-current events happens in a di/dt safe manner. The support is implemented in SOC_THERM (for pulse skipper) and in CAR (for PLL slowdown).

The intention of this section is not to provide a full programmer’s guide, but solely to document the register interface to allow better understanding of NVIDIA® provided drivers.

40.2 Thermal Sensor

Thermal and voltage sensors are used to constantly monitor the temperature and voltage on the chip. Sensors are placed across the die to gauge the temperature of the whole chip. The TSensor module:

- Generates interrupt to software to lower temperature via DVFS, on reaching a certain thermal/voltage threshold
- Generates signal to CAR to reduce CPU frequency by half, on reaching a certain thermal/voltage threshold
- Generates signal to PMC when temperature reaches dangerously high levels to reset the chip and sets a flag in PMC

40.3 TSensor Registers

Refer to “Reading Register Tables” in the Introduction section for the register table protocol as well as recommendations for accessing registers.

40.3.1 SOC_THERM_THERMCTL_LEVEL0_GROUP_CPU_0

Offset: 0x0 │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON

Bit	R/W	Reset	Description
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.2 SOC_THERM_THERMCTL_LEVEL0_GROUP_GPU_0

Offset: 0x4 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.3 SOC_THERM_THERMCTL_LEVEL0_GROUP_MEM_0

Offset: 0x8 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC

Bit	R/W	Reset	Description
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.4 SOC_THERM_THERMCTL_LEVEL0_GROUP_TSENSE_0

Offset: 0xc │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxx00000000000000000x00000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S: Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.5 SOC_THERM_THERMCTL_LEVEL0_UP_STATS_0

Offset: 0x10 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for UP threshold breaches for this level used in the lab

40.3.6 SOC_THERM_THERMCTL_LEVEL0_DN_STATS_0

Offset: 0x14 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for DN threshold breaches for this level used in the lab

40.3.7 SOC_THERM_THERMCTL_LEVEL1_GROUP_CPU_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.8 SOC_THERM_THERMCTL_LEVEL1_GROUP_GPU_0

Offset: 0x24 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY

Bit	R/W	Reset	Description
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.9 SOC_THERM_THERMCTL_LEVEL1_GROUP_MEM_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS:TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.10 SOC_THERM_THERMCTL_LEVEL1_GROUP_TSENSE_0

Offset: 0x2c | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000x00000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY

Bit	R/W	Reset	Description
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.11 SOC_THERM_THERMCTL_LEVEL1_UP_STATS_0

Offset: 0x30 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: count for UP threshold breaches for this level used in the lab

40.3.12 SOC_THERM_THERMCTL_LEVEL1_DN_STATS_0

Offset: 0x34 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: count for DN threshold breaches for this level used in the lab

40.3.13 SOC_THERM_THERMCTL_LEVEL2_GROUP_CPU_0

Offset: 0x40 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.14 SOC_THERM_THERMCTL_LEVEL2_GROUP_GPU_0

Offset: 0x44 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.15 SOC_THERM_THERMCTL_LEVEL2_GROUP_MEM_0

Offset: 0x48 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS:TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.16 SOC_THERM_THERMCTL_LEVEL2_GROUP_TSENSE_0

Offset: 0x4c | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000x00000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
6:5	RW	0x0	CPU:Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU:Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.17 SOC_THERM_THERMCTL_LEVEL2_UP_STATS_0

Offset: 0x50 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for UP threshold breaches for this level used in the lab

40.3.18 SOC_THERM_THERMCTL_LEVEL2_DN_STATS_0

Offset: 0x54 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for DN threshold breaches for this level used in the lab

40.3.19 SOC_THERM_THERMCTL_LEVEL3_GROUP_CPU_0

Offset: 0x60 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON

Bit	R/W	Reset	Description
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.20 SOC_THERM_THERMCTL_LEVEL3_GROUP_GPU_0

Offset: 0x64 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
7	RW	0x0	TS: TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.21 SOC_THERM_THERMCTL_LEVEL3_GROUP_MEM_0

Offset: 0x68 | Read/Write: R/W | Reset: 0x0000000X (0bxxxxxxx00000000000000000000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON

Bit	R/W	Reset	Description
7	RW	0x0	TS:TSENSE Modifier. Enable group only when TSENSE group is active 0 = OFF 1 = ON
6:5	RW	0x0	CPU :Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.22 SOC_THERM_THERMCTL_LEVEL3_GROUP_TSENSE_0

Offset: 0x6c │ Read/Write: R/W │ Reset: 0x0000000X (0bxxxxxxx00000000000000000x00000xx)

Bit	R/W	Reset	Description
24:17	RW	0x0	UP_THRESH: Threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
16:9	RW	0x0	DN_THRESH: Up threshold value for thermal sensor. Signed field for temperatures from -127ºC to +127ºC
8	RW	0x0	EN: Enable for temperature monitoring 0 = OFF 1 = ON
6:5	RW	0x0	CPU: Initiate CPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
4:3	RW	0x0	GPU: Initiate GPU throttling if level <N> is activated 0 = NONE 1 = LITE 2 = HEAVY
1:0	RO	X	S:Status 0 = BELLOW 1 = IN 2 = RES 3 = ABOVE

40.3.23 SOC_THERM_THERMCTL_LEVEL3_UP_STATS_0

Offset: 0x70 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT:count for UP threshold breaches for this level used in the lab

40.3.24 SOC_THERM_THERMCTL_LEVEL3_DN_STATS_0

Offset: 0x74 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Count for DN threshold breaches for this level used in the lab

40.3.25 SOC_THERM_THERMCTL_THERMTRIP_CTL_0

Offset: 0x80 | Read/Write: R/W | Reset: 0x10696969 (0bxxx10000011010010110100101101001)

Bit	Reset	Description
28	0x1	ANY_EN: Initiate THERMTRIP based on any monitoring group 0 = OFF 1 = ON
27	0x0	MEM_EN: Initiate THERMTRIP based on MEM monitoring group 0 = OFF 1 = ON
26	0x0	GPU_EN: Initiate THERMTRIP based on GPU monitoring group 0 = OFF 1 = ON
25	0x0	CPU_EN: Initiate THERMTRIP based on CPU monitoring group 0 = OFF 1 = ON
24	0x0	TSENSE_EN: Initiate THERMTRIP based on TSENSE monitoring group
23:16	0x69	GPU_N_MEM: Threshold for thermal shutdown for GPU group
15:8	0x69	CPU: Threshold for thermal shutdown for CPU group
7:0	0x69	TSENSE: Threshold for thermal shutdown for TSENSE group

40.3.26 SOC_THERM_THERMCTL_INTR_STATUS_0

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	MD3:MEM group down threshold interrupt for level 3 0 = OFF 1 = ON
30	0x0	MU3:MEM group up threshold interrupt for level 3 0 = OFF 1 = ON
29	0x0	MD2:MEM group down threshold interrupt for level 2 0 = OFF 1 = ON
28	0x0	MU2:MEM group up threshold interrupt for level 2 0 = OFF 1 = ON
27	0x0	MD1:MEM group down threshold interrupt for level 1 0 = OFF 1 = ON
26	0x0	MU1:MEM group up threshold interrupt for level 1 0 = OFF 1 = ON

Bit	Reset	Description
25	0x0	MD0:MEM group down threshold interrupt for level 0 0 = OFF 1 = ON
24	0x0	MU0:MEM group up threshold interrupt for level 0 0 = OFF 1 = ON
23	0x0	GD3:GPU group down threshold interrupt for level 3 0 = OFF 1 = ON
22	0x0	GU3:GPU group up threshold interrupt for level 3 0 = OFF 1 = ON
21	0x0	GD2:GPU group down threshold interrupt for level 2 0 = OFF 1 = ON
20	0x0	GU2:GPU group up threshold interrupt for level 2 0 = OFF 1 = ON
19	0x0	GD1:GPU group down threshold interrupt for level 1 0 = OFF 1 = ON
18	0x0	GU1:GPU group up threshold interrupt for level 1 0 = OFF 1 = ON
17	0x0	GD0:GPU group down threshold interrupt for level 0 0 = OFF 1 = ON
16	0x0	GU0:GPU group up threshold interrupt for level 0 0 = OFF 1 = ON
15	0x0	CD3:CPU group down threshold interrupt for level 3 0 = OFF 1 = ON
14	0x0	CU3:CPU group up threshold interrupt for level 3 0 = OFF 1 = ON
13	0x0	CD2:CPU group down threshold interrupt for level 2 0 = OFF 1 = ON
12	0x0	CU2:CPU group up threshold interrupt for level 2 0 = OFF 1 = ON
11	0x0	CD1:CPU group down threshold interrupt for level 1 0 = OFF 1 = ON
10	0x0	CU1:CPU group up threshold interrupt for level 1 0 = OFF 1 = ON
9	0x0	CD0:CPU group down threshold interrupt for level 0 0 = OFF 1 = ON

Bit	Reset	Description
8	0x0	CU0:CPU group up threshold interrupt for level 0 0 = OFF 1 = ON
7	0x0	TD3:TSENSE group down threshold interrupt for level 3 0 = OFF 1 = ON
6	0x0	TU3:TSENSE group up threshold interrupt for level 3 0 = OFF 1 = ON
5	0x0	TD2:TSENSE group down threshold interrupt for level 2 0 = OFF 1 = ON
4	0x0	TU2:TSENSE group up threshold interrupt for level 2 0 = OFF 1 = ON
3	0x0	TD1:TSENSE group down threshold interrupt for level 1 0 = OFF 1 = ON
2	0x0	TU1:TSENSE group up threshold interrupt for level 1 0 = OFF 1 = ON
1	0x0	TD0:TSENSE group down threshold interrupt for level 0 0 = OFF 1 = ON
0	0x0	TU0:TSENSE group up threshold interrupt for level 0 0 = OFF 1 = ON

40.3.27 SOC_THERM_THERMCTL_INTR_EN_0

Offset: 0x88 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	MD3:MEM group down threshold interrupt enable for level 3 0 = OFF 1 = ON
30	0x0	MU3:MEM group up threshold interrupt enable for level 3 0 = OFF 1 = ON
29	0x0	MD2:MEM group down threshold interrupt enable for level 2 0 = OFF 1 = ON
28	0x0	MU2:MEM group up threshold interrupt enable for level 2 0 = OFF 1 = ON
27	0x0	MD1:MEM group down threshold interrupt enable for level 1 0 = OFF 1 = ON
26	0x0	MU1:MEM group up threshold interrupt enable for level 1 0 = OFF 1 = ON
25	0x0	MD0:MEM group down threshold interrupt enable for level 0 0 = OFF 1 = ON

Bit	Reset	Description
24	0x0	MU0:MEM group up threshold interrupt enable for level 0 0 = OFF 1 = ON
23	0x0	GD3:GPU group down threshold interrupt enable for level 3 0 = OFF 1 = ON
22	0x0	GU3:GPU group up threshold interrupt enable for level 3 0 = OFF 1 = ON
21	0x0	GD2:GPU group down threshold interrupt enable for level 2 0 = OFF 1 = ON
20	0x0	GU2:GPU group up threshold interrupt enable for level 2 0 = OFF 1 = ON
19	0x0	GD1:GPU group down threshold interrupt enable for level 1 0 = OFF 1 = ON
18	0x0	GU1:GPU group up threshold interrupt enable for level 1 0 = OFF 1 = ON
17	0x0	GD0:GPU group down threshold interrupt enable for level 0 0 = OFF 1 = ON
16	0x0	GU0:GPU group up threshold interrupt enable for level 0 0 = OFF 1 = ON
15	0x0	CD3:CPU group down threshold interrupt enable for level 3 0 = OFF 1 = ON
14	0x0	CU3:CPU group up threshold interrupt enable for level 3 0 = OFF 1 = ON
13	0x0	CD2:CPU group down threshold interrupt enable for level 2 0 = OFF 1 = ON
12	0x0	CU2:CPU group up threshold interrupt enable for level 2 0 = OFF 1 = ON
11	0x0	CD1:CPU group down threshold interrupt enable for level 1 0 = OFF 1 = ON
10	0x0	CU1:CPU group up threshold interrupt enable for level 1 0 = OFF 1 = ON
9	0x0	CD0:CPU group down threshold interrupt enable for level 0 0 = OFF 1 = ON
8	0x0	CU0:CPU group up threshold interrupt enable for level 0 0 = OFF 1 = ON
7	0x0	TD3:TSENSE group down threshold interrupt enable for level 3 0 = OFF 1 = ON

Bit	Reset	Description
6	0x0	TU3:TSENSE group up threshold interrupt enable for level 3 0 = OFF 1 = ON
5	0x0	TD2:TSENSE group down threshold interrupt enable for level 2 0 = OFF 1 = ON
4	0x0	TU2:TSENSE group up threshold interrupt enable for level 2 0 = OFF 1 = ON
3	0x0	TD1:TSENSE group down threshold interrupt enable for level 1 0 = OFF 1 = ON
2	0x0	TU1:TSENSE group up threshold interrupt enable for level 1 0 = OFF 1 = ON
1	0x0	TD0:TSENSE group down threshold interrupt enable for level 0 0 = OFF 1 = ON
0	0x0	TU0:TSENSE group up threshold interrupt enable for level 0 0 = OFF 1 = ON

40.3.28 SOC_THERM_THERMCTL_INTR_DIS_0

Offset: 0x8c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31	0x0	MD3:MEM group down threshold interrupt disable for level 3 0 = OFF 1 = ON
30	0x0	MU3:MEM group up threshold interrupt disable for level 3 0 = OFF 1 = ON
29	0x0	MD2:MEM group down threshold interrupt disable for level 2 0 = OFF 1 = ON
28	0x0	MU2:MEM group up threshold interrupt disable for level 2 0 = OFF 1 = ON
27	0x0	MD1:MEM group down threshold interrupt disable for level 1 0 = OFF 1 = ON
26	0x0	MU1:MEM group up threshold interrupt disable for level 1 0 = OFF 1 = ON
25	0x0	MD0:MEM group down threshold interrupt disable for level 0 0 = OFF 1 = ON
24	0x0	MU0:MEM group up threshold interrupt disable for level 0 0 = OFF 1 = ON
23	0x0	GD3:GPU group down threshold interrupt disable for level 3 0 = OFF 1 = ON

Bit	Reset	Description
22	0x0	GU3:GPU group up threshold interrupt disable for level 3 0 = OFF 1 = ON
21	0x0	GD2:GPU group down threshold interrupt disable for level 2 0 = OFF 1 = ON
20	0x0	GU2:GPU group up threshold interrupt disable for level 2 0 = OFF 1 = ON
19	0x0	GD1:GPU group down threshold interrupt disable for level 1 0 = OFF 1 = ON
18	0x0	GU1:GPU group up threshold interrupt disable for level 1 0 = OFF 1 = ON
17	0x0	GD0:GPU group down threshold interrupt disable for level 0 0 = OFF 1 = ON
16	0x0	GU0:GPU group up threshold interrupt disable for level 0 0 = OFF 1 = ON
15	0x0	CD3:CPU group down threshold interrupt disable for level 3 0 = OFF 1 = ON
14	0x0	CU3:CPU group up threshold interrupt disable for level 3 0 = OFF 1 = ON
13	0x0	CD2:CPU group down threshold interrupt disable for level 2 0 = OFF 1 = ON
12	0x0	CU2:CPU group up threshold interrupt disable for level 2 0 = OFF 1 = ON
11	0x0	CD1:CPU group down threshold interrupt disable for level 1 0 = OFF 1 = ON
10	0x0	CU1:CPU group up threshold interrupt disable for level 1 0 = OFF 1 = ON
9	0x0	CD0:CPU group down threshold interrupt disable for level 0 0 = OFF 1 = ON
8	0x0	CU0:CPU group up threshold interrupt disable for level 0 0 = OFF 1 = ON
7	0x0	TD3:TSENSE group down threshold interrupt disable for level 3 0 = OFF 1 = ON
6	0x0	TU3:TSENSE group up threshold interrupt disable for level 3 0 = OFF 1 = ON
5	0x0	TD2:TSENSE group down threshold interrupt disable for level 2 0 = OFF 1 = ON

Bit	Reset	Description
4	0x0	TU2:TSENSE group up threshold interrupt disable for level 2 0 = OFF 1 = ON
3	0x0	TD1:TSENSE group down threshold interrupt disable for level 1 0 = OFF 1 = ON
2	0x0	TU1:TSENSE group up threshold interrupt disable for level 1 0 = OFF 1 = ON
1	0x0	TD0:TSENSE group down threshold interrupt disable for level 0 0 = OFF 1 = ON
0	0x0	TU0:TSENSE group up threshold interrupt disable for level 0 0 = OFF 1 = ON

40.3.29 SOC_THERM_THERMCTL_STATS_CTL_0

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0000)

Bit	Reset	Description
3	0x0	CLEAR_DN: Clear DN statistics for all levels
2	0x0	ENB_DN: Enable DN statistics collection for all levels
1	0x0	CLEAR_UP: Clear statistics for all levels
0	0x0	ENB_UP: Enable statistics collection for all levels

40.3.30 SOC_THERM_THERMCTL_SLOWDOWN_THRESHOLD_0

Offset: 0x98 | Read/Write: R/W | Reset: 0x69696969 (0b01101001011010010110100101101001)

Bit	Reset	Description
31:24	0x69	MEM: Slowdown threshold for MEM group
23:16	0x69	GPU: Slowdown threshold for GPU group
15:8	0x69	CPU: Slowdown threshold for CPU group
7:0	0x69	TSENSE: Slowdown threshold for TSENSE group

40.3.31 SOC_THERM_THERMCTL_SLOWDOWN_CTL_0

Offset: 0x9c | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxxxxxxxxxxxxxxx00000)

Bit	Reset	Description
31:30	0x0	SLOWDOWN_SELECT: Selects which one to throttle 0 = NONE 1 = CPU_ONLY 2 = GPU_ONLY 3 = BOTH
4	0x0	ANY_EN
3	0x0	MEM_EN
2	0x0	GPU_EN
1	0x0	CPU_EN

Bit	Reset	Description
0	0x0	TSENSE_EN

40.3.32 SOC_THERM_TSENSOR_CPU0_CONFIG0_0

Offset: 0xc0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: sensor stop 0 = OFF 1 = ON

40.3.33 SOC_THERM_TSENSOR_CPU0_CONFIG1_0

Offset: 0xc4 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.3.34 SOC_THERM_TSENSOR_CPU0_CONFIG2_0

Offset: 0xc8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.3.35 SOC_THERM_TSENSOR_CPU0_STATUS0_0

Offset: 0xcc | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: valid capture available
15:0	X	CAPTURE:last valid raw capture

40.3.36 SOC_THERM_TSENSOR_CPU0_STATUS1_0

Offset: 0xd0 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: valid capture available
15:0	X	TEMP: last valid translated temperature

40.3.37 SOC_THERM_TSENSOR_CPU0_STATUS2_0

Offset: 0xd4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.3.38 SOC_THERM_TSENSOR_CPU1_CONFIG0_0

Offset: 0xe0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

40.3.39 SOC_THERM_TSENSOR_CPU1_CONFIG1_0

Offset: 0xe4 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.3.40 SOC_THERM_TSENSOR_CPU1_CONFIG2_0

Offset: 0xe8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.3.41 SOC_THERM_TSENSOR_CPU1_STATUS0_0

Offset: 0xec | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

40.3.42 SOC_THERM_TSENSOR_CPU1_STATUS1_0

Offset: 0xf0 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.3.43 SOC_THERM_TSENSOR_CPU1_STATUS2_0

Offset: 0xf4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.3.44 SOC_THERM_TSENSOR_CPU2_CONFIG0_0

Offset: 0x100 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics

Bit	Reset	Description
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

40.3.45 SOC_THERM_TSENSOR_CPU2_CONFIG1_0

Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.3.46 SOC_THERM_TSENSOR_CPU2_CONFIG2_0

Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.3.47 SOC_THERM_TSENSOR_CPU2_STATUS0_0

Offset: 0x10c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID:valid capture available
15:0	X	CAPTURE:last valid raw capture

40.3.48 SOC_THERM_TSENSOR_CPU2_STATUS1_0

Offset: 0x110 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.3.49 SOC_THERM_TSENSOR_CPU2_STATUS2_0

Offset: 0x114 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.3.50 SOC_THERM_TSENSOR_CPU3_CONFIG0_0

Offset: 0x120 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

40.3.51 SOC_THERM_TSENSOR_CPU3_CONFIG1_0

Offset: 0x124 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.3.52 SOC_THERM_TSENSOR_CPU3_CONFIG2_0

Offset: 0x128 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.3.53 SOC_THERM_TSENSOR_CPU3_STATUS0_0

Offset: 0x12c │ Read/Write: RO │ Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

40.3.54 SOC_THERM_TSENSOR_CPU3_STATUS1_0

Offset: 0x130 │ Read/Write: RO │ Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.3.55 SOC_THERM_TSENSOR_CPU3_STATUS2_0

Offset: 0x134 │ Read/Write: RO │ Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.3.56 SOC_THERM_TSENSOR_MEM0_CONFIG0_0

Offset: 0x140 │ Read/Write: R/W │ Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: sensor stop 0 = OFF 1 = ON

40.3.57 SOC_THERM_TSENSOR_MEM0_CONFIG1_0

Offset: 0x144 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE:temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: time between en and capture
20:15	0x0	TIDDQ_EN: time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.3.58 SOC_THERM_TSENSOR_MEM0_CONFIG2_0

Offset: 0x148 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.3.59 SOC_THERM_TSENSOR_MEM0_STATUS0_0

Offset: 0x14c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

40.3.60 SOC_THERM_TSENSOR_MEM0_STATUS1_0

Offset: 0x150 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.3.61 SOC_THERM_TSENSOR_MEM0_STATUS2_0

Offset: 0x154 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.3.62 SOC_THERM_TSENSOR_MEM1_CONFIG0_0

Offset: 0x160 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics

Bit	Reset	Description
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

40.3.63 SOC_THERM_TSENSOR_MEM1_CONFIG1_0

Offset: 0x164 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.3.64 SOC_THERM_TSENSOR_MEM1_CONFIG2_0

Offset: 0x168 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.3.65 SOC_THERM_TSENSOR_MEM1_STATUS0_0

Offset: 0x16c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

40.3.66 SOC_THERM_TSENSOR_MEM1_STATUS1_0

Offset: 0x170 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.3.67 SOC_THERM_TSENSOR_MEM1_STATUS2_0

Offset: 0x174 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.3.68 SOC_THERM_TSENSOR_GPU_CONFIG0_0

Offset: 0x180 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

40.3.69 SOC_THERM_TSENSOR_GPU_CONFIG1_0

Offset: 0x184 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: Time between en and capture
20:15	0x0	TIDDQ_EN: Time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.3.70 SOC_THERM_TSENSOR_GPU_CONFIG2_0

Offset: 0x188 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.3.71 SOC_THERM_TSENSOR_GPU_STATUS0_0

Offset: 0x18c | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

40.3.72 SOC_THERM_TSENSOR_GPU_STATUS1_0

Offset: 0x190 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.3.73 SOC_THERM_TSENSOR_GPU_STATUS2_0

Offset: 0x194 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.3.74 SOC_THERM_TSENSOR_PLLX_CONFIG0_0

Offset: 0x1a0 | Read/Write: R/W | Reset: 0x00000001 (0bxxxx00000000000000000xx000001)

Bit	Reset	Description
27:8	0x0	TALL: M count
5	0x0	STATUS_CLR: Clears MIN/MAX statistics
4	0x0	TCALC_OVERFLOW: Add operation overflow 0 = OFF 1 = ON
3	0x0	OVERFLOW: 14-bit overflow 0 = OFF 1 = ON
2	0x0	CPTR_OVERFLOW: Capture overflow 0 = OFF 1 = ON
1	0x0	RO_SEL: Ring oscillator select 0 = TS 1 = VS
0	0x1	STOP: Sensor stop 0 = OFF 1 = ON

40.3.75 SOC_THERM_TSENSOR_PLLX_CONFIG1_0

Offset: 0x1a4 | Read/Write: R/W | Reset: 0x00000000 (0b0x000000xxx000000xxxxx0000000000)

Bit	Reset	Description
31	0x0	TEMP_ENABLE: Temperature enable 0 = OFF 1 = ON
29:24	0x0	TEN_COUNT: time between en and capture
20:15	0x0	TIDDQ_EN: time between IDDQ and en
9:0	0x0	TSAMPLE: N count

40.3.76 SOC_THERM_TSENSOR_PLLX_CONFIG2_0

Offset: 0x1a8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:16	0x0	THERM_A:
15:0	0x0	THERM_B:

40.3.77 SOC_THERM_TSENSOR_PLLX_STATUS0_0

Offset: 0x1ac | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	CAPTURE_VALID: Valid capture available
15:0	X	CAPTURE: Last valid raw capture

40.3.78 SOC_THERM_TSENSOR_PLLX_STATUS1_0

Offset: 0x1b0 | Read/Write: RO | Reset: 0xX000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31	X	TEMP_VALID: Valid capture available
15:0	X	TEMP: Last valid translated temperature

40.3.79 SOC_THERM_TSENSOR_PLLX_STATUS2_0

Offset: 0x1b4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	TEMP_MAX: Maximum temperature registered since last clear, reset to lowest possible temperature
15:0	X	TEMP_MIN: Minimum temperature registered since last clear, reset to highest possible temperature

40.3.80 SOC_THERM_TSENSOR_PDIV_0

Offset: 0x1c0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
15:12	0x0	CPU_PDIV: PDIV for TS_CPU0 TS_CPU3
11:8	0x0	GPU_PDIV: PDIV for TS_GPU

Bit	Reset	Description
7:4	0x0	MEM_PDIV: PDIV for TS_MEM
3:0	0x0	PLLX_PDIV: PDIV for TS_PLLX

40.3.81 SOC_THERM_TSENSOR_HOTSPOT_OFF_0

Offset: 0x1c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx000000000000000000000000)

Bit	Reset	Description
23:16	0x0	CPU_HOTSPOT_OFF:CPU hotspot offset from PLLX
15:8	0x0	GPU_HOTSPOT_OFF:GPU hotspot offset from PLLX
7:0	0x0	MEM_HOTSPOT_OFF:MEM hotspot offset from PLLX

40.3.82 SOC_THERM_TSENSOR_TEMP1_0

Offset: 0x1c8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	CPU_TEMP: Processed CPU temperature seen by thermal throttling logic. Temp readback format
15:0	X	GPU_TEMP: Processed GPU temperature seen by thermal throttling logic. Temp readback format

40.3.83 SOC_THERM_TSENSOR_TEMP2_0

Offset: 0x1cc | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	MEM_TEMP: Processed MEM temperature seen by thermal throttling logic. Temperature readback format
15:0	X	SENSOR_TEMP: Processed sensor (PLLX) temperature seen by thermal throttling logic. Temperature readback format

40.3.84 SOC_THERM_TSENSOR_TSENSOR_PWR_VLD_OVERRIDE_0

Offset: 0x1d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0)

Bit	Reset	Description
0	0x0	INVALIDATE_ON_PWR_GATING: Use CPU/GPU virtual power gating status to invalidate the CPU/GPU sensors if set otherwise use Rail gating status

40.3.85 SOC_THERM_TSENSOR_SPARE_ECO_0

Offset: 0x1d4 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	Reserved.

40.3.86 SOC_THERM_EDP_OC_ALARM_OC1_CFG_0

Offset: 0x310 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000100)

Bit	Reset	Description
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

40.3.87 SOC_THERM_EDP_OC_ALARM_OC1_CNT_THRESHOLD_0

Offset: 0x314 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

40.3.88 SOC_THERM_EDP_OC_ALARM_OC1_THROTTLE_PERIOD_0

Offset: 0x318 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds. Valid only for brief throttle

40.3.89 SOC_THERM_EDP_OC_ALARM_OC1_COUNT_0

Offset: 0x31c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Event count

40.3.90 SOC_THERM_EDP_OC_ALARM_OC1_FILTER_0

Offset: 0x320 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: Event count

40.3.91 SOC_THERM_EDP_OC_ALARM_OC2_CFG_0

Offset: 0x324 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000100)

Bit	Reset	Description
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

40.3.92 SOC_THERM_EDP_OC_ALARM_OC2_CNT_THRESHOLD_0

Offset: 0x328 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

40.3.93 SOC_THERM_EDP_OC_ALARM_OC2_THROTTLE_PERIOD_0

Offset: 0x32c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds. Valid only for brief throttle

40.3.94 SOC_THERM_EDP_OC_ALARM_OC2_COUNT_0

Offset: 0x330 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Event count

40.3.95 SOC_THERM_EDP_OC_ALARM_OC2_FILTER_0

Offset: 0x334 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: Event count

40.3.96 SOC_THERM_EDP_OC_ALARM_OC3_CFG_0

Offset: 0x338 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000100)

Bit	Reset	Description
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

40.3.97 SOC_THERM_EDP_OC_ALARM_OC3_CNT_THRESHOLD_0

Offset: 0x33c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

40.3.98 SOC_THERM_EDP_OC_ALARM_OC3_THROTTLE_PERIOD_0

Offset: 0x340 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds- valid only for brief throttle

40.3.99 SOC_THERM_EDP_OC_ALARM_OC3_COUNT_0

Offset: 0x344 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	COUNT: Event count

40.3.100 SOC_THERM_EDP_OC_ALARM_OC3_FILTER_0

Offset: 0x348 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: Event count

40.3.101 SOC_THERM_EDP_OC_ALARM_OC4_CFG_0

Offset: 0x34c | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000100)

Bit	Reset	Description
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

40.3.102 SOC_THERM_EDP_OC_ALARM_OC4_CNT_THRESHOLD_0

Offset: 0x350 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

40.3.103 SOC_THERM_EDP_OC_ALARM_OC4_THROTTLE_PERIOD_0

Offset: 0x354 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds- valid only for brief throttle

40.3.104 SOC_THERM_EDP_OC_ALARM_OC4_COUNT_0

Offset: 0x358 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Event count

40.3.105 SOC_THERM_EDP_OC_ALARM_OC4_FILTER_0

Offset: 0x35c | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: Event count

40.3.106 SOC_THERM_EDP_OC_ALARM_OC5_CFG_0

Offset: 0x360 | Read/Write: R/W | Reset: 0x00000004 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxx0000100)

Bit	Reset	Description
6	0x0	LONG_LATENCY_THROTTLE: Used to inform the CAR PLLx hardware controller that hardware is expected to control the PLL for large amounts of time when this OC alarm is engaged
5	0x0	HW_RESTORE_EN: Auto-restore system to unthrottled state after this alarm is deasserted 0 = DISABLE 1 = ENABLE
4	0x0	PWRGOOD_MASK: Mask the throttling if power is not good. 0 = ENABLE_THROTTLE 1 = DISABLE_THROTTLE
3:2	0x1	THROTTLE_MODE: To select sticky versus brief mode throttling 0 = DISABLED 1 = STICKY 2 = BRIEF 3 = RESERVED
1	0x0	ALARM_POLARITY:1-assert low, 0-assert high
0	0x0	EN_THROTTLE: Enable the throttling on this over current alarm

40.3.107 SOC_THERM_EDP_OC_ALARM_OC5_CNT_THRESHOLD_0

Offset: 0x364 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

40.3.108 SOC_THERM_EDP_OC_ALARM_OC5_THROTTLE_PERIOD_0

Offset: 0x368 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	OC_THROTTLE_PERIOD: Brief throttle period in microseconds- valid only for brief throttle

40.3.109 SOC_THERM_EDP_OC_ALARM_OC5_COUNT_0

Offset: 0x36c | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	COUNT: Event count

40.3.110 SOC_THERM_EDP_OC_ALARM_OC5_FILTER_0

Offset: 0x370 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: Event count

40.3.111 SOC_THERM_EDP_OC_INTR_STATUS_0

Offset: 0x39c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx0xx00xxxxxxxxxx00000)

Bit	Reset	Description
20	0x0	CLDVFS_DIDT:CLDVFS interrupt status
4	0x0	OC5: OC event 5 interrupt status
3	0x0	OC4: OC event 4 interrupt status
2	0x0	OC3: OC event 3 interrupt status
1	0x0	OC2: OC event 2 interrupt status
0	0x0	OC1: OC event 1 interrupt status

40.3.112 SOC_THERM_EDP_OC_INTR_ENABLE_0

Offset: 0x3a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx0xx00xxxxxxxxxx00000)

Bit	Reset	Description
20	0x0	CLDVFS_DIDT_EN:CLDVFS interrupt enable
4	0x0	OC5_EN: OC event 5 interrupt enable
3	0x0	OC4_EN: OC event 4 interrupt enable
2	0x0	OC3_EN: OC event 3 interrupt enable
1	0x0	OC2_EN: OC event 2 interrupt enable
0	0x0	OC1_EN: OC event 1 interrupt enable

40.3.113 SOC_THERM_EDP_OC_INTR_DISABLE_0

Offset: 0x3a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxx0xx00xxxxxxxxxx00000)

Bit	Reset	Description
20	0x0	CLDVFS_DIDT_DIS:CLDVFS interrupt disable
4	0x0	OC5_DIS: OC event 5 interrupt disable
3	0x0	OC4_DIS: OC event 4 interrupt disable
2	0x0	OC3_DIS: OC event 3 interrupt disable
1	0x0	OC2_DIS: OC event 2 interrupt disable
0	0x0	OC1_DIS: OC event 1 interrupt disable

40.3.114 SOC_THERM_EDP_OC_ALARM_OC1_STATS_0

Offset: 0x3a8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

40.3.115 SOC_THERM_EDP_OC_ALARM_OC2_STATS_0

Offset: 0x3ac | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

40.3.116 SOC_THERM_EDP_OC_ALARM_OC3_STATS_0

Offset: 0x3b0 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

40.3.117 SOC_THERM_EDP_OC_ALARM_OC4_STATS_0

Offset: 0x3b4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

40.3.118 SOC_THERM_EDP_OC_ALARM_OC5_STATS_0

Offset: 0x3b8 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

40.3.119 SOC_THERM_EDP_OC_ALARM_STATS_CTRL_0

Offset: 0x3c4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00)

Bit	Reset	Description
1	0x0	CLEAR_ALL: Clear all statistics
0	0x0	ENB_ALL: Enable all statistics collections for all counters

40.3.120 SOC_THERM_EDP_OC_CLDVFS_DIDT_CNT_THRESHOLD_0

Offset: 0x3c8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	EVENT_CNT_THRESHOLD: Event count threshold to raise interrupt, 0-rise interrupt for every event

40.3.121 SOC_THERM_EDP_OC_CLDVFS_DIDT_EVENT_COUNT_0

Offset: 0x3cc | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	COUNT: Event count

40.3.122 SOC_THERM_EDP_OC_CLDVFS_DIDT_EVENT_CNT_FILTER_0

Offset: 0x3d0 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	COUNT: event count

40.3.123 SOC_THERM_EDP_OC_CLDVFS_DIDT_EVENT_STATS_0

Offset: 0x3d4 | Read/Write: RO | Reset: 0xXXXXXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	EVENT_COUNT: Statistics for event count

40.3.124 SOC_THERM_EDP_OC_ALARM_THROTTLE_PERIOD_CTL_0

Offset: 0x3d8 | Read/Write: R/W | Reset: 0x00000088 (0bxxxxxxxxxxxxxxxxxxxxxxxx10001000)

Bit	Reset	Description
7:0	0x88	NUM_CLKS_IN_1US: Number of soc_therm clocks in 1µs used to determine brief throttle length (default based on 136 MHz)

40.3.125 SOC_THERM_THROTTLECTL_GLOBAL_THROTTLE_CFG_0

Offset: 0x400 | Read/Write: R/W | Reset: 0x00000001 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0001)

Bit	Reset	Description
3	0x0	DFLL_PSKIP_CTRL:0: (default) pause CPU pulse skipper when cldvfs2soc_therm_skipper1_en is asserted1: do not pause CPU pulse skipper
2	0x0	PSKIP_RESTORE_CTRL:0: Software does not restore (default) 1: Software restores. If HW_RESTORE_EN==0, hardware will not restore the pulse skipper
1	0x0	SW_OVERRIDE_MODE: Pulse skipper software override :=1 causes all throttle indicators to use software pulse skipper cfg for throttling pulse skipper 0 = DISABLE 1 = ENABLE
0	0x1	ENB: Single bit to disable all throttling 0 = DISABLE 1 = ENABLE

40.3.126 SOC_THERM_THROTTLECTL_SW_CPU_PSKIP_CTRL_0

Offset: 0x408 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.127 SOC_THERM_THROTTLECTL_SW_CPU_PSKIP_RAMP_RATE_0

Offset: 0x40c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.128 SOC_THERM_THROTTLECTL_SW_GPU_PSKIP_CTRL_0

Offset: 0x410 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.129 SOC_THERM_THROTTLECTL_SW_GPU_PSKIP_RAMP_RATE_0

Offset: 0x414 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLED 1 = ENABLED
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.130 SOC_THERM_THROTTLECTL_CPU_PSKIP_STATUS_0

Offset: 0x418 | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:12	X	CURR_PULSE_SKIP_M:
11:4	X	CURR_PULSE_SKIP_N:
1	X	SW_OVERRIDE_STATUS: 0 = DISABLED 1 = ENABLED
0	X	ENB_STATUS: 0 = DISABLED 1 = ENABLED

40.3.131 SOC_THERM_THROTTLECTL_GPU_PSKIP_STATUS_0

Offset: 0x41c | Read/Write: RO | Reset: 0x000XXXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:12	X	CURR_PULSE_SKIP_M:
11:4	X	CURR_PULSE_SKIP_N:
1	X	SW_OVERRIDE_STATUS: 0 = DISABLED 1 = ENABLED
0	X	ENB_STATUS: 0 = DISABLED 1 = ENABLED

40.3.132 SOC_THERM_THROTTLECTL_PRIORITY_LOCK_0

Offset: 0x424 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Maximum priority allowed for software programmable vectors

40.3.133 SOC_THERM_THROTTLECTL_THROTTLE_STATUS_0

Offset: 0x428 | Read/Write: RO | Reset: 0x0000XXXX (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
12	X	PRIORITY_LOCK_BREACH: Set to '1' if the selected throttle vector is of higher priority than the limit specified by the boot loader. The field is cleared when software updates PRIORITY registers.
11:4	X	THROTTLE_SEQ_STATE:0=WAIT
0	X	ENB_STATUS: Global enable status 0 = DISABLED 1 = ENABLED

40.3.134 SOC_THERM_THROTTLECTL_LITE_CPU_PSKIP_CTRL_0

Offset: 0x430 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.135 SOC_THERM_THROTTLECTL_LITE_CPU_PSKIP_RAMP_RATE_0

Offset: 0x434 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.136 SOC_THERM_THROTTLECTL_LITE_GPU_PSKIP_CTRL_0

Offset: 0x438 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.137 SOC_THERM_THROTTLECTL_LITE_GPU_PSKIP_RAMP_RATE_0

Offset: 0x43c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.138 SOC_THERM_THROTTLECTL_LITE_THROTTLE_PRIORITY_0

Offset: 0x444 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

40.3.139 SOC_THERM_THROTTLECTL_LITE_THROTTLE_DELAY_0

Offset: 0x448 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

40.3.140 SOC_THERM_THROTTLECTL_HEAVY_CPU_PSKIP_CTRL_0

Offset: 0x460 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.141 SOC_THERM_THROTTLECTL_HEAVY_CPU_PSKIP_RAMP_RATE_0

Offset: 0x464 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.142 SOC_THERM_THROTTLECTL_HEAVY_GPU_PSKIP_CTRL_0

Offset: 0x468 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.143 SOC_THERM_THROTTLECTL_HEAVY_GPU_PSKIP_RAMP_RATE_0

Offset: 0x46c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.144 SOC_THERM_THROTTLECTL_HEAVY_THROTTLE_PRIORITY_0

Offset: 0x474 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

40.3.145 SOC_THERM_THROTTLECTL_HEAVY_THROTTLE_DELAY_0

Offset: 0x478 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

40.3.146 SOC_THERM_THROTTLECTL_OC1_CPU_PSKIP_CTRL_0

Offset: 0x490 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.147 SOC_THERM_THROTTLECTL_OC1_CPU_PSKIP_RAMP_RATE_0

Offset: 0x494 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.148 SOC_THERM_THROTTLECTL_OC1_GPU_PSKIP_CTRL_0

Offset: 0x498 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.149 SOC_THERM_THROTTLECTL_OC1_GPU_PSKIP_RAMP_RATE_0

Offset: 0x49c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.150 SOC_THERM_THROTTLECTL_OC1_THROTTLE_PRIORITY_0

Offset: 0x4a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

40.3.151 SOC_THERM_THROTTLECTL_OC1_THROTTLE_DELAY_0

Offset: 0x4a8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

40.3.152 SOC_THERM_THROTTLECTL_OC2_CPU_PSKIP_CTRL_0

Offset: 0x4c0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.153 SOC_THERM_THROTTLECTL_OC2_CPU_PSKIP_RAMP_RATE_0

Offset: 0x4c4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.154 SOC_THERM_THROTTLECTL_OC2_GPU_PSKIP_CTRL_0

Offset: 0x4c8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.155 SOC_THERM_THROTTLECTL_OC2_GPU_PSKIP_RAMP_RATE_0

Offset: 0x4cc | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.156 SOC_THERM_THROTTLECTL_OC2_THROTTLE_PRIORITY_0

Offset: 0x4d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

40.3.157 SOC_THERM_THROTTLECTL_OC2_THROTTLE_DELAY_0

Offset: 0x4d8 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

40.3.158 SOC_THERM_THROTTLECTL_OC3_CPU_PSKIP_CTRL_0

Offset: 0x4f0 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.159 SOC_THERM_THROTTLECTL_OC3_CPU_PSKIP_RAMP_RATE_0

Offset: 0x4f4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.160 SOC_THERM_THROTTLECTL_OC3_GPU_PSKIP_CTRL_0

Offset: 0x4f8 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.161 SOC_THERM_THROTTLECTL_OC3_GPU_PSKIP_RAMP_RATE_0

Offset: 0x4fc | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.162 SOC_THERM_THROTTLECTL_OC3_THROTTLE_PRIORITY_0

Offset: 0x504 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

40.3.163 SOC_THERM_THROTTLECTL_OC3_THROTTLE_DELAY_0

Offset: 0x508 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

40.3.164 SOC_THERM_THROTTLECTL_OC4_CPU_PSKIP_CTRL_0

Offset: 0x520 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.165 SOC_THERM_THROTTLECTL_OC4_CPU_PSKIP_RAMP_RATE_0

Offset: 0x524 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.166 SOC_THERM_THROTTLECTL_OC4_GPU_PSKIP_CTRL_0

Offset: 0x528 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.167 SOC_THERM_THROTTLECTL_OC4_GPU_PSKIP_RAMP_RATE_0

Offset: 0x52c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx000000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.168 SOC_THERM_THROTTLECTL_OC4_THROTTLE_PRIORITY_0

Offset: 0x534 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

40.3.169 SOC_THERM_THROTTLECTL_OC4_THROTTLE_DELAY_0

Offset: 0x538 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000000000000)

Bit	Reset	Description
31:0	0x0	DELAY_VAL: Number of soc_therm_clk cycles to delay the alert

40.3.170 SOC_THERM_THROTTLECTL_OC5_CPU_PSKIP_CTRL_0

Offset: 0x550 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.171 SOC_THERM_THROTTLECTL_OC5_CPU_PSKIP_RAMP_RATE_0

Offset: 0x554 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.172 SOC_THERM_THROTTLECTL_OC5_GPU_PSKIP_CTRL_0

Offset: 0x558 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxxxxxxxxxx0000000000000000)

Bit	Reset	Description
31	0x0	ENB: Enable the throttling 0 = DISABLE 1 = ENABLE
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = m+1
7:0	0x0	SUPER_CDIV_DIVISOR: Actual Value = n+1

40.3.173 SOC_THERM_THROTTLECTL_OC5_GPU_PSKIP_RAMP_RATE_0

Offset: 0x55c | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx00000000000000000000000)

Bit	Reset	Description
31	0x0	SEQ_BYPASS_MODE: Bypass throttle seq and directly apply settings 0 = DISABLE 1 = ENABLE
23:8	0x0	PULSE_SKIP_DURATION: Duration in soc_therm_clk cycles
7:0	0x0	PULSE_SKIP_STEP: Pulse skip step size, 0:+/-1

40.3.174 SOC_THERM_THROTTLECTL_OC5_THROTTLE_PRIORITY_0

Offset: 0x564 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
7:0	0x0	PRIORITY: Bigger value indicates higher priority. In general, higher priority translates to lower target frequency

[THIS PAGE INTENTIONALLY LEFT BLANK]

Notice

The information provided in this specification is believed to be accurate and reliable as of the date provided. However, NVIDIA Corporation (“NVIDIA”) does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This publication supersedes and replaces all other specifications for the product that may have been previously supplied.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and other changes to this specification, at any time and/or to discontinue any product or service without notice. Customer should obtain the latest relevant specification before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer. NVIDIA hereby expressly objects to applying any customer general terms and conditions with regard to the purchase of the NVIDIA product referenced in this specification.

NVIDIA products are not designed, authorized or warranted to be suitable for use in medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on these specifications will be suitable for any specified use without further testing or modification. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to ensure the product is suitable and fit for the application planned by customer and to do the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this specification. NVIDIA does not accept any liability related to any default, damage, costs or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this specification, or (ii) customer product designs.

No license, either express or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this specification. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA. Reproduction of information in this specification is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without alteration, and is accompanied by all associated conditions, limitations, and notices.

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA’s aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the NVIDIA terms and conditions of sale for the product.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

ARM

ARM, AMBA and ARM Powered are registered trademarks of ARM Limited. Cortex, MPCore and Mali are trademarks of ARM Limited. All other brands or product names are the property of their respective holders. "ARM" is used to represent ARM Holdings plc; its operating company ARM Limited; and the regional subsidiaries ARM Inc.; ARM KK; ARM Korea Limited.; ARM Taiwan Limited; ARM France SAS; ARM Consulting (Shanghai) Co. Ltd.; ARM Germany GmbH; ARM Embedded Technologies Pvt. Ltd.; ARM Norway, AS and ARM Sweden AB.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Trademarks

NVIDIA, the NVIDIA logo, and Tegra are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright