TECHNICAL REFERENCE MANUAL

NVIDIA TEGRA 3
HD Mobile Processors

NVIDIA® Tegra® 3 HD Mobile Processors: T30 Series, AP30 Series

Abstract

The Technical Reference Manual focuses on the logical organization and control of NVIDIA® Tegra® 3 devices. 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	JAN 27, 2012	Public Release to support Open Source Development. The following sections are undergoing further review and should be considered preliminary: • CLOCK AND RESET CONTROLLER • MULTI-PURPOSE I/O PINS AND PIN MULTIPLEXING (PINMUXING) • POWER MANAGEMENT CONTROLLER
v02p	MAR 18, 2013	The following changes were made in this revision: • Added VI, GR2D, and EPP sections • Clock and Reset Controller: Removed references to unsupported clocking functions. • CPU: Changed timer divider rate to divide by 2. • Display Controller: Changed A_ADDR_V_OFFSET field definition and MAX value in DC_WIN_A_DDA_INCREMENT_0 calculations. Removed references to unsupported display modes. FIFO depth increased to 128 in “Interface to Memory”. • GMI Controller: In Programming Guidelines, added bullet about DMA completion in interrupt polling mode. • SPI Controller: Revised the APB DMA requester numbers for the SPI controllers. Updated clock frequencies in “Clock Initialization and Control”. • UART and VFIR Controller: Updated “Hardware Signaling”.
v03p	SEP 12, 2013	The following changes were made in this revision: • Added FLOW CONTROLLER section

Table of Contents

1.0 INTRODUCTION

NVIDIA® Tegra® 3 processors are complete applications and digital media systems built around these processing elements:

- CPU Complex: Quad Cortex®-A9 Symmetric Multi-Processing ARM® Cores with vSMP. The Cortex-A9 core features dual 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 1 MB shared Level 2 cache. The NVIDIA vSMP architecture adds a fifth Companion Core which operates exclusively with the main CPU complex, and allows low-power low-leakage operation at lighter CPU loads.
- Audio/Video Decoder: the AVP (Audio-Video Processor) subsystem includes dedicated audio and video decode hardware acceleration, an ARM7 processor, and embedded RAM. This subsystem provides full motion playback of up to 1080P high-definition video and supports the H.264, VC-1, and MPEG-4 video standards and multiple audio standards with dedicated hardware.
- Video Encoder: A high performance H.264 1080P capable hardware video encoder. This processor supports H.264 Baseline Profile Encoding and MPEG4 Simple Profile Encoding.
- Graphics: Ultra Low-power NVIDIA GeForce® Graphics Processing Unit (GPU). Handles 2D graphics rendering and 3D pixel and vertex shading. Dual pixel rendering pipes.
- Imaging: A high quality hardware accelerated still-image and video capture path supporting Bayer and YUV format sensors.
- Display: Dual display controllers with various output options for LCD panels and televisions, including HDMI output up to 1080p.

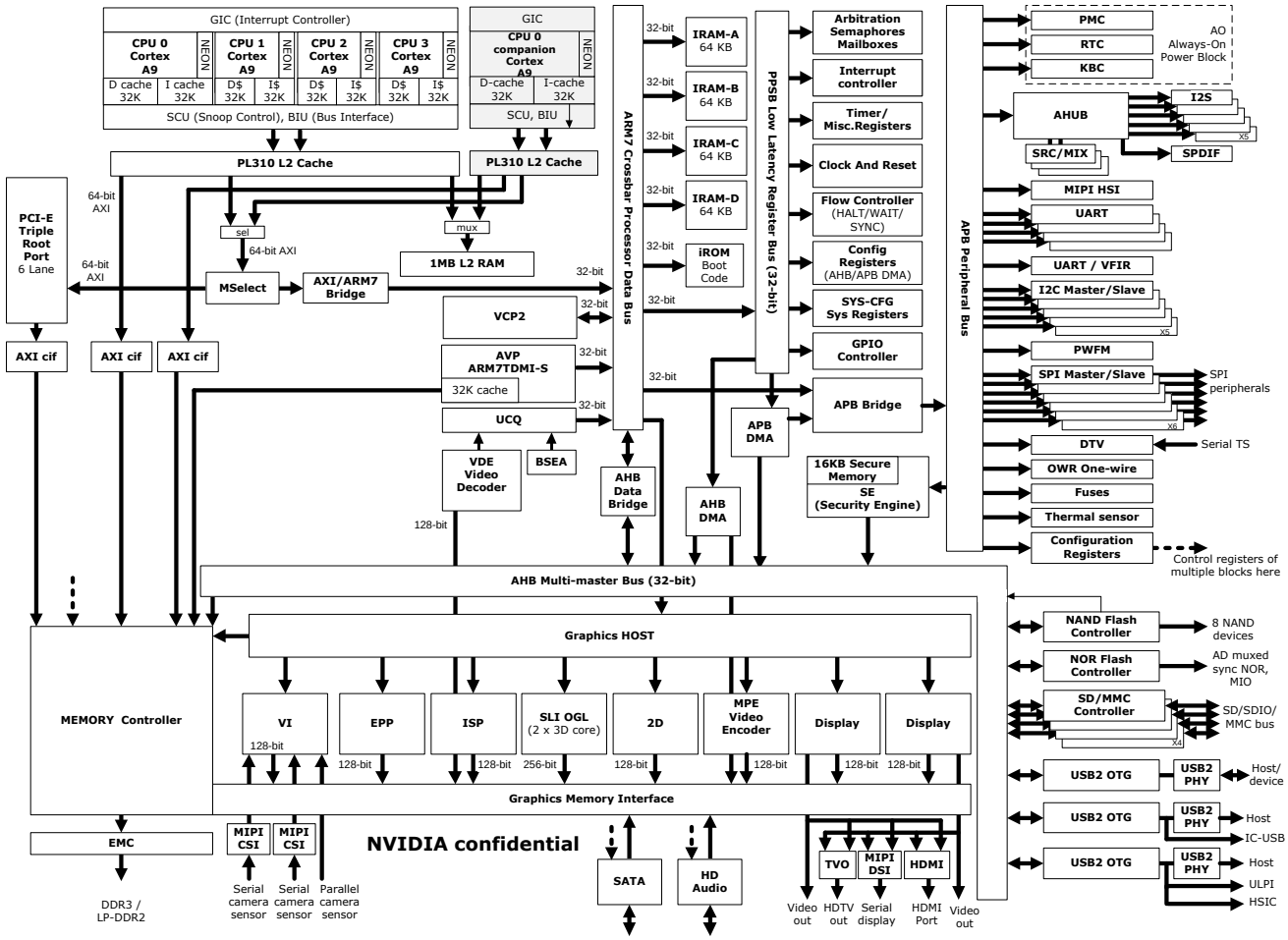
In addition to its processing elements, Tegra 3 has a broad range of peripheral interfaces to enable communication with wireless baseband, other communications peripherals, audio codecs, power management, and mass storage. When combined with baseband and PMU chips, the Tegra 3 processor provides the functionality needed to build a smart-phone or tablet. Dedicated high-performance mass storage controllers, with their own DMA engines, free the CPU Complex from routine data management tasks.

Intended as the heart of a low-powered portable multimedia device, the Tegra 3 device integrates a quad-core ARM Cortex-A9 MPCore™ processor to run the operating system, user interface, and required software applications. The video and audio acceleration hardware provide a highly optimized low-power system for 2D and 3D graphics rendering, audio and video recording and playback, video conferencing, high-quality image capture and display.

1.1 Block Diagram

This diagram provides an overview of a Tegra 3 processor.

Figure 1. Tegra 3 Block Diagram



1.2 Memory Controller and Internal Bus Architecture

The Tegra 3 device has a highly optimized 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 services by a shared multi-channel APB DMA controller which is also an APB device.

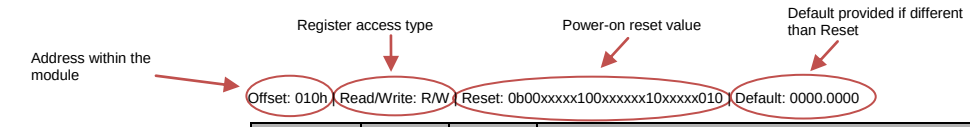
Special provision is made for the CPU to bypass much of the memory controller arbitration with low latency.

1.3 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:** is the address of the register within the specific module. Refer the system memory map for the start address of the module, 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
- **Default:** will only be displayed if the default setting is different from the Reset value.

Unspecified bits may not appear in tables (see example below). Writes to unspecified bits are ignored, while reads return an unknown value.



Unspecified bits in this example register: 32:26, 23:19, 15:10, 7:3

Unspecified bits are shown as 'x' in the Reset field above the table.

Leading bits that are unspecified may not be displayed in the Reset value. In this example register, the leading bits (32:26) are unspecified and not shown in the Reset field. For example

Reset:
0bxxxxxxx00xxxxx100xxxxxx10xxxxx010
vs.
0b00xxxxx100xxxxxx10xxxxx010

Bit	R/W	Reset	Description
25:24	RW	N1	EMEM_NUMDEV 0 = N1 1 = N2
18:16	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
2:0	RO	W9	EMEM_COLWIDTH 0 = W7 1 = W8 2 = W9 3 = W10 4 = W11

1.4 Glossary

Term	Definition
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.
AMBA™	Advanced Microcontroller Bus Architecture, a set of standard buses defined by ARM®.
APB	AMBA Peripheral Bus, a simple 32-bit single master bus for peripheral devices.
AVP	Audio-Video Processor, the term used both to describe the ARM7 processor in Tegra 3 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 Tegra 3.
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 Tegra 3.
CVE	Zoran TV Encoder
DSI	MIPI Display Serial Interface, a standard high-speed serial interface for connecting displays to Tegra 3.
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 Tegra 3 capable of filtering, color-space conversion, rotation and pixel format conversion.
HDMI	High-Definition Multimedia Interface, a digital connection carrying uncompressed video and audio at high speed over a single connector.
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
KBC	Keyboard Controller module allows Tegra 3 to be connected to keyboard matrices of sizes up to 16x8.
MC	Memory Controller module handles requests from internal clients and arbitrates among them to allocate memory bandwidth.
MIPI	The Mobile Industry Processor Interface and industry alliance promoting a number of standard interfaces for mobile devices.
MPE	Video Encoder in Tegra 3 capable of encoding raw video stream into MPEG
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 3 devices and accelerated in hardware by dedicated 3D and 2D engines.
PMC	Power Management Controller module controls the various power management features in the system.
PWFM	Pulse Width Frequency Modulation module generates programmed pulse widths typically used to control backlight in display panels.
PVT	Process, Voltage & Temperature

Term	Definition
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
TWC	Three-Wire Controller
TVO	TV Encoder Output
UCQ	Unified Command Queue – a sub module within Video Decoder Engine
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, the Tegra 3 hardware acceleration block dedicated to decoding compressed video in various formats.
VI	Video Input block, the acronym used to describe the Tegra 3 block used for camera and related input functions.

2.0 ADDRESS AND INTERRUPT MAP

2.1 System Memory Map

Table 1. System Memory Map

Description	Address Start	Address End	Default Length
PCIE Memory (CPU only)	**0000:0000**	**3fff:ffff**	**1 GB**
PCIE	0000:0000	3fff:ffff	1 GB
Data Memory	**4000:0000**	**47ff:ffff**	**128 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
NOR Flash	**4800:0000**	**4fff:ffff**	**128 MB**
NOR Flash	4800:0000	4fff:ffff	128 MB
Graphics Host Registers	**5000:0000**	**5002:3fff**	**144 KB**
Host1x	5000:0000	5002:3fff	144 KB
ARM Cortex®-A9 registers (@ PERIPHBASE)	**5004:0000**	**5004:1fff**	**8 KB**
ARM PERIPHBASE	5004:0000	5004:1fff	8 KB
ARM Interrupt Distributor	5004:1000	5004:1fff	4 KB
AVP CACHE	5004:0000	5004:1fff	8 KB
MSelect	**5004:2000**	**5004:2fff**	**4 KB**
MSelect Register Space	5004:2000	5004:2fff	4 KB
PL310 (L2 Cache)	**5004:3000**	**5004:3fff**	**4 KB**
ARM PL310	5004:3000	5004:3fff	4 KB
Verif Aperture	**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
BBM0	5303:3000	5303:30ff	256 B
BBM1	5303:3100	5303:31ff	256 B
BBM2	5303:3200	5303:32ff	256 B
Graphics Host	**5400:0000**	**57ff:ffff**	**64 MB**
MPE	5404:0000	5407:ffff	256 KB
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

Description	Address Start	Address End	Default Length
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
GART (legacy aperture only)	**5800:0000**	**58ff:ffff**	**16 MB**
	5800:0000	59ff:ffff	32 MB
PPSB Bus device registers	**6000:0000**	**67ff:ffff**	**128 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
TMR	6000:5000	6000:53ff	1 KB
TMR1			8 B
TMR2			8 B
TMRUS			64 B
TMR3			8 B
TMR4			8 B
TMR5			8 B
TMR6			8 B
TMR7			8 B
TMR8			8 B
TMR9			8 B
TMR0			8 B
WDT0			32 B
WDT1			32 B
WDT2			32 B
WDT3			32 B
WDT4			32 B
TMR_SHARED			32 B
CLK 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
External I/O	**6800:0000**	**6fff:ffff**	**128 MB**
MIO/EXIO	6800:0000	6fff:ffff	128 MB
APB	**7000:0000**	**77ff:ffff**	**128 MB**
MISC	7000:0000	7000:3fff	16 KB
PP			1 KB
SC1X_PADS			1 KB
GP			1 KB
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	7000:6400	7000:64ff	256 B
HDMI_IOBIST	7000:6500	7000:65ff	256 B
MIPI_IOBIST	7000:6600	7000:66ff	256 B
LPDDR2_IOBIST	7000:6700	7000:67ff	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
NAND Controller	7000:8000	7000:81ff	512 B
HSMMC/CE-ATA	7000:8500	7000:85ff	256 B
Sync NOR	7000:9000	7000:9fff	4 KB
PWFM Controller	7000:a000	7000:a0ff	256 B
HSI Baseband	7000:b000	7000:b0ff	256 B
I2C	7000:c000	7000:c0ff	256 B
TWC	7000:c100	7000:c1ff	256 B

Description	Address Start	Address End	Default Length
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
SLINK 2B-1	7000:d400	7000:d5ff	512 B
SLINK 2B-2	7000:d600	7000:d7ff	512 B
SLINK 2B-3	7000:d800	7000:d9ff	512 B
SLINK 2B-4	7000:da00	7000:dbff	512 B
SLINK 2B-5	7000:dc00	7000:ddff	512 B
SLINK 2B-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:e7ff	1 KB
MC	7000:f000	7000:f3ff	1 KB
EMC	7000:f400	7000:f7ff	1 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
SATA	7002:0000	7002:ffff	64 KB
HDA	7003:0000	7003:ffff	64 KB
CoreSight™ Debug Registers (CSITE)	7004:0000	7007:ffff	256 KB
AUDIO_CLUSTER	7008:0000	7008:ffff	64 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
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
AHB	**7800:0000**	**7fff:ffff**	**128 MB**

Description	Address Start	Address End	Default Length
PPCS (AHB to MC flush)	7c00:0000	7c00:ffff	64 KB
TZRAM	7c01:0000	7c01:ffff	64 KB
USB	7d00:0000	7d00:3fff	16 KB
USB2	7d00:4000	7d00:7fff	16 KB
USB3	7d00:8000	7d00:bfff	16 KB
SDMMC-1	7800:0000	7800:01ff	512 B
SDMMC-2	7800:0200	7800:03ff	512 B
SDMMC-3	7800:0400	7800:05ff	512 B
SDMMC-4	7800:0600	7800:07ff	512 B
External Memory	**8000:0000**	**ffef:ffff**	**2047 MB**
EMEM	8000:0000	ffef:ffff	2047 MB
IROM (boot code)	**fff0:0000**	**fff0:bfff**	**48 KB**
Hi-VEC	**ffff:0000**	**ffff:ffff**	**64 KB**

2.2 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 Cortex-A9 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 2. Primary Interrupt Controller Mapping

Block Name	Interrupt Controller	Interrupt Number
TMR1	PRI_ICTLR	0
TMR2	PRI_ICTLR	1
RTC	PRI_ICTLR	2
CEC	PRI_ICTLR	3
SHR-SEM	PRI_ICTLR	4
SHR-SEM	PRI_ICTLR	5
SHR-SEM	PRI_ICTLR	6
SHR-SEM	PRI_ICTLR	7
VDE UCQ Error Interrupt	PRI_ICTLR	8
VDE Sync Token Interrupt	PRI_ICTLR	9
VDE BSE-V Interrupt	PRI_ICTLR	10
VDE BSE-A Interrupt	PRI_ICTLR	11
VDE SXE Interrupt	PRI_ICTLR	12
SATA_RX_STAT	PRI_ICTLR	13
SDMMC1	PRI_ICTLR	14
SDMMC2	PRI_ICTLR	15
XIO	PRI_ICTLR	16
VDE Interrupt	PRI_ICTLR	17
AVP_UCQ	PRI_ICTLR	18
SDMMC3	PRI_ICTLR	19
USB	PRI_ICTLR	20
USB2	PRI_ICTLR	21
HSMMC	PRI_ICTLR	22
SATA_CTL	PRI_ICTLR	23
NANDFLASH	PRI_ICTLR	24
VCP	PRI_ICTLR	25
APB_DMA	PRI_ICTLR	26
AHB_DMA	PRI_ICTLR	27
GNT.1 Arbitration Grant Status of COP	PRI_ICTLR	28
GNT.0 Arbitration Grant Status of CPU	PRI_ICTLR	29
OWR	PRI_ICTLR	30

Block Name	Interrupt Controller	Interrupt Number
SDMMC4	PRI_ICTLR	31

Table 3. Secondary Interrupt Controller Mapping

Block Name	Interrupt Controller	Interrupt Number
GPIO1	SEC_ICTLR	0
GPIO2	SEC_ICTLR	1
GPIO3	SEC_ICTLR	2
GPIO4	SEC_ICTLR	3
UARTA	SEC_ICTLR	4
UARTB	SEC_ICTLR	5
I2C	SEC_ICTLR	6
SPI	SEC_ICTLR	7
TWC	SEC_ICTLR	8
TMR3	SEC_ICTLR	9
TMR4	SEC_ICTLR	10
FLOW - RSM0 Resume for CPU	SEC_ICTLR	11
FLOW - RSM1 Resume for COP	SEC_ICTLR	12
ACTMON	SEC_ICTLR	13
UARTC	SEC_ICTLR	14
MIPI baseband controller	SEC_ICTLR	15
EVENT	SEC_ICTLR	16
EVENT	SEC_ICTLR	17
EVENT	SEC_ICTLR	18
EVENT	SEC_ICTLR	19
VFIR	SEC_ICTLR	20
I2C5	SEC_ICTLR	21
SYS_STATS_MON	SEC_ICTLR	22
GPIO5	SEC_ICTLR	23
SPEEDO_PMON_0	SEC_ICTLR	24
SPEEDO_PMON_1	SEC_ICTLR	25
SE	SEC_ICTLR	26
SLINK (SBC1)	SEC_ICTLR	27
APB_DMA_COP	SEC_ICTLR	28
AHB_DMA_COP	SEC_ICTLR	29
DMA TX Interrupt	SEC_ICTLR	30
DMA RX Interrupt	SEC_ICTLR	31

Table 4. Tertiary Interrupt Controller Mapping

Block Name	Interrupt Controller	Interrupt Number
HOST1X - COP SyncPt Interrupt	TRI_ICTLR	0
HOST1X - MPCORE SyncPt Interrupt	TRI_ICTLR	1
HOST1X - COP General Interrupt	TRI_ICTLR	2
HOST1X - MPCORE General Interrupt	TRI_ICTLR	3
MPE General Interrupt	TRI_ICTLR	4
VI General Interrupt	TRI_ICTLR	5
EPP General Interrupt	TRI_ICTLR	6
ISP General Interrupt	TRI_ICTLR	7
2D General Interrupt	TRI_ICTLR	8
Display General Interrupt	TRI_ICTLR	9
Display B General Interrupt	TRI_ICTLR	10
HDMI INT from pins	TRI_ICTLR	11
TVO General Interrupt	TRI_ICTLR	12
MC General Interrupt	TRI_ICTLR	13
EMC General Interrupt	TRI_ICTLR	14
SLINK-6 (SBC6)	TRI_ICTLR	15
NOR FLASH (Misc)	TRI_ICTLR	16
HDA	TRI_ICTLR	17
SLINK-2 (SBC2)	TRI_ICTLR	18
SLINK-3 (SBC3)	TRI_ICTLR	19
I2C2	TRI_ICTLR	20
KBC	TRI_ICTLR	21
External PMU	TRI_ICTLR	22
GPIO6	TRI_ICTLR	23
TVDAC (from MISC)	TRI_ICTLR	24
GPIO7	TRI_ICTLR	25
UARTD	TRI_ICTLR	26
UARTE	TRI_ICTLR	27
I2C3	TRI_ICTLR	28
SLINK-4 (SBC4)	TRI_ICTLR	29
SLINK-5 (SBC5)	TRI_ICTLR	30
SW Reserved Interrupt	TRI_ICTLR	31

Table 5. Quaternary Interrupt Controller Mapping

Block Name	Interrupt Controller	Interrupt Number
SNOR	QUAD_ICTLR	0
USB3	QUAD_ICTLR	1
PCIE	QUAD_ICTLR	2
PCIE	QUAD_ICTLR	3
PCIE	QUAD_ICTLR	4
AVP_CACHE	QUAD_ICTLR	5
TSENSOR	QUAD_ICTLR	6
AUDIO_CLUSTER	QUAD_ICTLR	7
APB_DMA_CH0	QUAD_ICTLR	8
APB_DMA_CH1	QUAD_ICTLR	9
APB_DMA_CH2	QUAD_ICTLR	10
APB_DMA_CH3	QUAD_ICTLR	11
APB_DMA_CH4	QUAD_ICTLR	12
APB_DMA_CH5	QUAD_ICTLR	13
APB_DMA_CH6	QUAD_ICTLR	14
APB_DMA_CH7	QUAD_ICTLR	15
APB_DMA_CH8	QUAD_ICTLR	16
APB_DMA_CH9	QUAD_ICTLR	17
APB_DMA_CH10	QUAD_ICTLR	18
APB_DMA_CH11	QUAD_ICTLR	19
APB_DMA_CH12	QUAD_ICTLR	20
APB_DMA_CH13	QUAD_ICTLR	21
APB_DMA_CH14	QUAD_ICTLR	22
APB_DMA_CH15	QUAD_ICTLR	23
I2C4	QUAD_ICTLR	24
TMR5	QUAD_ICTLR	25
TMR_SHARED	QUAD_ICTLR	26
WDT_CPU	QUAD_ICTLR	27
WDT_AVP	QUAD_ICTLR	28
GPIO8	QUAD_ICTLR	29
CAR	QUAD_ICTLR	30
<NONE>	QUAD_ICTLR	31

Table 6. Fifth Interrupt Controller Mapping

Block Name	Interrupt Controller	Interrupt Number
APB_DMA_CH16	PENTA_ICTLR	0
APB_DMA_CH17	PENTA_ICTLR	1
APB_DMA_CH18	PENTA_ICTLR	2
APB_DMA_CH19	PENTA_ICTLR	3
APB_DMA_CH20	PENTA_ICTLR	4
APB_DMA_CH21	PENTA_ICTLR	5
APB_DMA_CH22	PENTA_ICTLR	6
APB_DMA_CH23	PENTA_ICTLR	7
APB_DMA_CH24	PENTA_ICTLR	8
APB_DMA_CH25	PENTA_ICTLR	9
APB_DMA_CH26	PENTA_ICTLR	10
APB_DMA_CH27	PENTA_ICTLR	11
APB_DMA_CH28	PENTA_ICTLR	12
APB_DMA_CH29	PENTA_ICTLR	13
APB_DMA_CH30	PENTA_ICTLR	14
APB_DMA_CH31	PENTA_ICTLR	15
CPU0_PMU_INTR	PENTA_ICTLR	16
CPU1_PMU_INTR	PENTA_ICTLR	17
CPU2_PMU_INTR	PENTA_ICTLR	18
CPU3_PMU_INTR	PENTA_ICTLR	19
CPU4_PMU_INTR	PENTA_ICTLR	20
CPU5_PMU_INTR	PENTA_ICTLR	21
CPU6_PMU_INTR	PENTA_ICTLR	22
CPU7_PMU_INTR	PENTA_ICTLR	23
<NONE>	PENTA_ICTLR	24
<NONE>	PENTA_ICTLR	25
<NONE>	PENTA_ICTLR	26
<NONE>	PENTA_ICTLR	27
<NONE>	PENTA_ICTLR	28
<NONE>	PENTA_ICTLR	29
<NONE>	PENTA_ICTLR	30
<NONE>	PENTA_ICTLR	31

2.3 APB DMA Interrupts

The APB DMA controller has a second level interrupt controller where the interrupts from the various channels are merged together to form the DMA TX and DMA RX interrupts shown above.

Table 7. APB DMA TX Interrupt Mapping

Block Name	Interrupt Controller	Interrupt Number
	DMA_TX_INTR	0
APBIF_CH0	DMA_TX_INTR	1
APBIF_CH1	DMA_TX_INTR	2
APBIF_CH2	DMA_TX_INTR	3
APBIF_CH3	DMA_TX_INTR	4
	DMA_TX_INTR	5
	DMA_TX_INTR	6
SPI	DMA_TX_INTR	7
UART-A	DMA_TX_INTR	8
UART-B	DMA_TX_INTR	9
UART-C	DMA_TX_INTR	10
TWC	DMA_TX_INTR	11
I2C	DMA_TX_INTR	12
I2C2	DMA_TX_INTR	13
UART-D	DMA_TX_INTR	14
UART-E	DMA_TX_INTR	15
I2C3	DMA_TX_INTR	16
VFIR	DMA_TX_INTR	17
I2C4	DMA_TX_INTR	18
I2C5	DMA_TX_INTR	19
<unused>	DMA_TX_INTR	20
<unused>	DMA_TX_INTR	21
<unused>	DMA_TX_INTR	22
<unused>	DMA_TX_INTR	23
<unused>	DMA_TX_INTR	24
<unused>	DMA_TX_INTR	25
<unused>	DMA_TX_INTR	26
<unused>	DMA_TX_INTR	27
<unused>	DMA_TX_INTR	28
<unused>	DMA_TX_INTR	29
<unused>	DMA_TX_INTR	30
<unused>	DMA_TX_INTR	31

Table 8. APB DMA RX Interrupt Mapping

Block Name	Interrupt Controller	Interrupt Number
	DMA_RX_INTR	0
APBIF_CH0	DMA_RX_INTR	1
APBIF_CH1	DMA_RX_INTR	2
APBIF_CH2	DMA_RX_INTR	3
APBIF_CH3	DMA_RX_INTR	4
	DMA_RX_INTR	5
	DMA_RX_INTR	6
SPI	DMA_RX_INTR	7
UART-A	DMA_RX_INTR	8
UART-B	DMA_RX_INTR	9
UART-C	DMA_RX_INTR	10
TWC	DMA_RX_INTR	11
I2C RX Slave	DMA_RX_INTR	12
I2C2	DMA_RX_INTR	13
UART-D	DMA_RX_INTR	14
UART-E	DMA_RX_INTR	15
I2C3	DMA_RX_INTR	16
VFIR	DMA_RX_INTR	17
I2C4	DMA_RX_INTR	18
I2C5	DMA_RX_INTR	19
<unused>	DMA_RX_INTR	20
<unused>	DMA_RX_INTR	21
<unused>	DMA_RX_INTR	22
<unused>	DMA_RX_INTR	23
<unused>	DMA_RX_INTR	24
<unused>	DMA_RX_INTR	25
<unused>	DMA_RX_INTR	26
<unused>	DMA_RX_INTR	27
<unused>	DMA_RX_INTR	28
<unused>	DMA_RX_INTR	29
<unused>	DMA_RX_INTR	30
<unused>	DMA_RX_INTR	31

[THIS PAGE INTENTIONALLY LEFT BLANK]

3.0 INTERRUPT CONTROLLER

The section describes the “legacy” interrupt controller, which is used for the AVP and is a backup option for the ARM® Cortex®-A9 CPU. The CPU’s local interrupt controller, the GIC, should be used in preference to the legacy interrupt controller because of its SMP support, and fast register access due to being local to the CPU.

The Tegra® 3 device has five sets of interrupt controllers, each handling 32 possible interrupt sources. A flexible interrupt controller allows priority scheduling and Interrupt Request steering. This allows any interrupt request to be re-routed to either processor to allow dynamic redistribution of workload, resulting in increased efficiency.

The interrupt controllers act as a centralized distribution center. There are two levels of interrupt priorities, Fast Interrupt Request (FIQ) and Regular Interrupt Request (IRQ). Each flag is evaluated by software to determine whether it gets the FIQ or IRQ. Generally, interrupts which require low latency get the FIQ. IRQ status is given to the remaining interrupts. Interrupt enable can mask or allow requests. The CPU and AVP (COP) can set priorities for valid interrupts. The ARM processor supports two types of hardware interrupts (nIRQ and nFIQ). The nIRQ signal is the normal active-LOW interrupt signal. The nFIQ signal is a fast interrupt signal that is used to manage time critical or short tasks such as USB transfer requests. Any of the interrupt requests can be routed to either the nIRQ or the nFIQ of either processor, based on the select bits set in the interrupt class and interrupt enable registers.

Interrupt enabling and steering is accomplished by programming the Interrupt Enable registers (CPU_IER or COP_IER) and the Interrupt Class registers (CPU_IEP_CLASS or COP_IEP_CLASS). The following text discusses routing as it relates to the CPU, but this discussion also applies to the AVP (COP).

When a 1 is set in the proper bit position in the CPU_IER register, that particular source is capable of interrupting the processor. The interrupt status register (ISR) allows the processor to view the state of the pending interrupt requests, whether enabled or disabled. The forced interrupt status register (FIR) allows the software to selectively force the execution of a specific interrupt service routine. The read-only VFIQ allows each processor to determine the actual source of the interrupt request(s) causing the processor to enter the interrupt service routine.

Tegra 3 devices can also use software DMA to handle DMA requests. The DMA interrupt requests are grouped into read requests and write requests and presented at bits 31 and 30 of the ISR register. Individual requests can be viewed by reading the DSR register. The Arbitration Semaphore module can also generate interrupt requests to the interrupt controller when a processor achieves Arbitration Grant Status. When a 1 is set in the proper bit position of the Arbitration Semaphore Interrupt Source Enable register (GNT0_CPU.ENABLE), that particular request causes an interrupt when the Grant Status becomes true.

3.1 Functionality

The programmable Interrupt Controller supports up to 160 interrupts and is built from five 32-bit interrupt controllers – Primary (PRI), Secondary (SEC), Tertiary (TRI), Quaternary (QUAD), and Quinary (PENTA).

Each interrupt request is treated individually, each with its own Interrupt_Enable bit (IER), Interrupt_Class steering bit (IEP_CLASS), and software forced interrupt bit (FIR). Each processor has its own set of IER and IEP_Class registers, so an interrupt can be steered to either (or both) processor as IRQ or FIQ.

When a 1 is set the proper bit position of the CPU_IER register, that particular interrupt source (ISR) is allowed to interrupt the processor. Whether it is sent to the processor as an IRQ or FIQ depends on the bit setting of the CPU_IEP_CLASS register. A '1' causes the interrupt to be routed to the processor nFIQ pin, a '0' to the nIRQ pin.

All the individual CPU IRQ bits are ORed together to form the final CPU IRQ.

All the individual CPU FIQ bits are ORed together to form the final CPU FIQ.

All the individual AVP (COP) IRQ bits are ORed together to form the final AVP (COP) IRQ.

All the individual AVP (COP) IRQ bits are ORed together to form the final AVP (COP) IRQ.

The individual interrupt bits are also output as a bus to be routed to the Exception Vector logic.

The CPU_IER register bits are SET by writing to the CPU_IER_SET register. Writing a '1' in individual bit positions will only alter those bits. The CPU_IER register bits are cleared by writing to the CPU_IER_CLR register. The same applies to the COP_IER registers. Using the SET/CLR methods avoids problems caused by two processors trying to perform Read-Modify-Write operations on respective bits at the same time. For testing, software can force individual interrupts by setting bits in the FIR register. These bits are ORed with the incoming interrupt to form the final interrupt for each bit position.

The Secondary Interrupt controller also supports Soft DMA. Since the APB_DMA controller has a limited number of DMA channels, it may not always be possible for every module to use hardware DMA. In this case the DMA request can be routed to the interrupt controller and treated like an interrupt. A software based DMA interrupt handler can then service the interrupt. This is especially useful for devices which are not latency sensitive. The Secondary interrupt controller has support for Soft DMA. There are 32 RX DRQ request inputs (DRX_RX_SOURCES), and 32 TX DRQ request inputs (DRX_TX_SOURCES). Masking for each request is programmed via the DRQ_RX_ENABLE and DRQ_TX_ENABLE registers. All the qualified DRQ_RX interrupts are ORed together and presented as Interrupt Request bit 31 input to the Primary interrupt controller. All the qualified DRQ_TX interrupts are ORed together and presented as Interrupt Request bit 30 input to the Primary interrupt controller. The DRQ_RX_STATUS and DRQ_TX_STATUS registers indicates DRQ status after masking.

The Primary Interrupt controller also supports Arbitration Grant interrupts. The Arbitration Semaphore module provides a mechanism for Processors or Processes to arbitrate for hardware or software resources. 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 controller can then interrupt the processor to tell it that the Grant occurred. The Primary Interrupt control supports 32 bits of ARM.GNT sources. The ARB_GNT_ENABLE register controls the masking on the request. The ARB_GNT_STATUS register indicates status after masking. The OR of all the individual ARB_GNT bits is presented to the Primary Interrupt controller as IRQ bit 29.

3.1.1 Interrupt Function Mapping

The mapping of interrupts to the bits defined in this interrupt controller is described elsewhere in this document.

3.2 Interrupt Registers

The system has four sets of Interrupt Controllers, for a total of 128 bits. There are two levels of interrupt priority, Fast Interrupt Request (FIQ), and Regular Interrupt Request (IRQ). Any of the interrupt requests can be routed to either the nIRQ or the nFIQ of either processor, based on the select bits set in the interrupt class and interrupt enable registers.

Interrupt enabling and steering is accomplished by programming the Interrupt Enable registers (CPU_IER or COP_IER) and the Interrupt Class registers (CPU_IEP_CLASS or COP_IEP_CLASS). A discussion on routing as it relates to the CPU follows, but this is also applicable to the COP (AVP).

When a 1 is set in the proper bit position in the CPU_IER register, that particular source is capable of interrupting the processor. The interrupt status register (ISR) allows the processor to view the state of the pending interrupt requests, whether enabled or disabled. The forced interrupt status register (FIR) allows the software to selectively force the execution of a specific interrupt service routine.

The read-only VIRQ_CPU and VIRQ_COP registers allow the processor to determine the actual source of the interrupt request(s) causing the processor to enter the nIRQ interrupt service routine. The VIRQ is the logical OR of the FIR and ISR registers, ANDed with the CPU_IER register, and ANDed with the NOT of the CPU_IEP_CLASS register.

The read-only VFIQ allows the processor to determine the actual source of the interrupt request(s) causing the processor to enter the nFIQ interrupt service routine. The VFIQ is the logical OR of the FIR and ISR registers, ANDed with the CPU_IER register, and ANDed with the CPU_IEP_CLASS register.

The CPU_IER and FIR registers also have corresponding set and clear registers, which allow bits to be turned on or off in a single atomic operation.

It is also possible to perform software DMA for many modules, as the DMA requests are also routed to the interrupt controller so they can be satisfied by interrupt handlers.

The DMA requests are grouped into read requests and write requests and presented at bits 31 and 30 of the ISR register (Secondary interrupt controller only). Individual requests can be viewed by reading the DSR register.

The Arbitration Semaphore module can also generate interrupt requests to the interrupt controller when a processor achieves Arbitration Grant Status.

3.2.1 Primary Interrupt Controller Registers

3.2.1.1 PRI_ICTLR_VIRQ_CPU_0

Valid Interrupt Request Status for CPU Register

IRQ31_IRQ0: Flags set by Hardware, cleared by Software.

Offset: 000h | Read/Write: RO | Reset: 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
23	X	SATA_CTL
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	SATA_RX_STAT
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

3.2.1.2 PRI_ICTLR_VIRQ_COP_0

Valid Interrupt Status for AVP (COP) Register

IRQ31_IRQ0: Flags set by Hardware, cleared by Software

Offset: 004h | Read/Write: RO | Reset: 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
23	X	SATA_CTL
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ

Bit	Reset	Description
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	SATA_RX_STAT
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

3.2.1.3 PRI_ICTLR_VFIQ_CPU_0

Valid Interrupt Status for CPU Register

FIQ31_FIQ0: Flags set by Hardware, cleared by Software.

Offset: 008h | Read/Write: RO | Reset: 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

Bit	Reset	Description
23	X	SATA_CTL
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	SATA_RX_STAT
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

3.2.1.4 PRI_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for AVP (COP) Register

FIQ31_FIQ0: Flags set by Hardware, cleared by Software.

Offset: 00ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SDMMC4
30	X	OWR

Bit	Reset	Description
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
23	X	SATA_CTL
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	SATA_RX_STAT
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

3.2.1.5 PRI_ICTLR_ISR_0

Latched Interrupt Status Register (Hardware)

ISR31_ISR0: Read-only. Set by hardware event, cleared at source by software.

Offset: 010h | Read/Write: RO | Reset: 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
23	X	SATA_CTL
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	SATA_RX_STAT
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

3.2.1.6 PRI_ICTLR_FIR_0

Forced Interrupt Status Register (Software)

FIR31_FIR0: Read only: Set during write to FIR_SET, cleared during write to FIR_CLR.

Offset: 014h | Read/Write: RO | Reset: 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
23	X	SATA_CTL
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	SATA_RX_STAT
12	X	VDE_SXE
11	X	VDE_BSEA
10	X	VDE_BSEV

Bit	Reset	Description
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

3.2.1.7 PRI_ICTLR_FIR_SET_0

Force Interrupt Register Set

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

Offset: 018h | Read/Write: WO | Reset: 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
23	0x0	SATA_CTL
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO

Bit	Reset	Description
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	SATA_RX_STAT
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

3.2.1.8 PRI_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

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

Offset: 01ch | Read/Write: WO | Reset: 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
23	0x0	SATA_CTL
22	0x0	SDMMC4

Bit	Reset	Description
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	SATA_RX_STAT
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

3.2.1.9 PRI_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

IER31_IER0: Interrupt Enable Status. 0 = Disabled.

Offset: 020h | Read/Write: RO | Reset: 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
23	X	SATA_CTL
22	X	SDMMC4
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	SATA_RX_STAT
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

3.2.1.10 PRI_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

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

Offset: 024h | Read/Write: WO | Reset: 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
23	0x0	SATA_CTL
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	SATA_RX_STAT
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

3.2.1.11 PRI_ICTLR_CPU_IER_CLR_0

CPU Clear Interrupt Enable for CPU

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

Offset: 028h | Read/Write: WO | Reset: 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
23	0x0	SATA_CTL
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	SATA_RX_STAT
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

3.2.1.12 PRI_ICTLR_CPU_IEP_CLASS_0

CPU Interrupt Enable Priority Class (FIQ/IRQ)

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

Offset: 02ch | Read/Write: R/W | Reset: 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
23	0x0	SATA_CTL
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO

Bit	Reset	Description
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	SATA_RX_STAT
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

3.2.1.13 PRI_ICTLR_COP_IER_0

Enabled Interrupt Source for AVP (COP) Register

IER31_IER0: Interrupt Enable Status. 0 = Disabled.

Offset: 030h | Read/Write: RO | Reset: 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
23	X	SATA_CTL
22	X	SDMMC4

Bit	Reset	Description
21	X	USB2
20	X	USB
19	X	SDMMC3
18	X	AVP_UCQ
17	X	VDE
16	X	XIO
15	X	SDMMC2
14	X	SDMMC1
13	X	SATA_RX_STAT
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

3.2.1.14 PRI_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

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

Offset: 034h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SDMMC4
30	0x0	OWR
29	0x0	ARB_SEM_GNT_CPU
28	0x0	ARB_SEM_GNT_COP

Bit	Reset	Description
27	0x0	AHB_DMA_CPU
26	0x0	APB_DMA_CPU
25	0x0	VCP
24	0x0	NANDCTRL
23	0x0	SATA_CTL
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	SATA_RX_STAT
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

3.2.1.15 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 the COP.

Offset: 038h | Read/Write: WO | Reset: 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
23	0x0	SATA_CTL
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	SATA_RX_STAT
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

3.2.1.16 PRI_ICTLR_COP_IEP_CLASS_0

COP Interrupt Enable Priority Class (FIQ/IRQ) Register

Set priority interrupt source for the AVP (COP). 1 = FIQ. 0 = IRQ.

Offset: 03ch | Read/Write: R/W | Reset: 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
23	0x0	SATA_CTL
22	0x0	SDMMC4
21	0x0	USB2
20	0x0	USB
19	0x0	SDMMC3
18	0x0	AVP_UCQ
17	0x0	VDE
16	0x0	XIO
15	0x0	SDMMC2
14	0x0	SDMMC1
13	0x0	SATA_RX_STAT
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 Arbitration Semaphore Interrupt Registers

3.2.2.1 PRI_ICTLR_ARBGNT_CPU_STATUS_0

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.

CPU Arbitration Semaphore Interrupt Status Register

Offset: 040h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

3.2.2.2 PRI_ICTLR_ARBGNT_CPU_ENABLE_0

CPU Arbitration Semaphore Interrupt Enable Register

Offset: 044h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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

3.2.2.3 PRI_ICTLR_ARBGNT_COP_STATUS_0

COP Arbitration Semaphore Interrupt Status Register

Offset: 048h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

3.2.2.4 PRI_ICTLR_ARBGNT_COP_ENABLE_0

COP Arbitration Semaphore Interrupt Enable Register

Offset: 04ch | Read/Write: R/W | Reset: 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 Second Interrupt Controller

Second interrupt controller base registers.

3.2.3.1 SEC_ICTLR_VIRQ_CPU_0

Valid Interrupt Request Status for CPU Register

IRQ31_IRQ0: Flags set by Hardware, cleared by Software.

Offset: 100h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	SPEEDO_PMON_1
24	X	SPEEDO_PMON_0
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C

Bit	Reset	Description
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	DTV
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.3.2 SEC_ICTLR_VIRQ_COP_0

Valid Interrupt Status for COP Register

IRQ31_IRQ0: Flags set by Hardware, cleared by Software.

Offset: 104h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	SPEEDO_PMON_1
24	X	SPEEDO_PMON_0
23	X	GPIO5
22	X	STAT_MON

Bit	Reset	Description
21	X	I2C5
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	DTV
6	X	I2C
5	X	UARTB
4	X	UARTC
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.3.3 SEC_ICTLR_VFIQ_CPU_0

FIQ Valid Interrupt Status for CPU Register

FIQ31_FIQ0: Flags set by Hardware, cleared by Software.

Offset: 108h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	SPEEDO_PMON_1
24	X	SPEEDO_PMON_0
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	DTV
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4

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

3.2.3.4 SEC_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for COP Register

FIQ31_FIQ0: Flags set by Hardware, cleared by Software.

Offset: 10ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	SPEEDO_PMON_1
24	X	SPEEDO_PMON_0
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	DTV

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

3.2.3.5 SEC_ICTLR_ISR_0

Latched Interrupt Status Register (HW)

ISR31_ISR0: Set by hardware event, cleared at source by software.

Offset: 110h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	SPEEDO_PMON_1
24	X	SPEEDO_PMON_0
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU

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

3.2.3.6 SEC_ICTLR_FIR_0

Forced Interrupt Status Register (SW)

FIR31_FIR0: Set during write to FIR_SET, cleared during write to FIR_CLR.

Offset: 114h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	SPEEDO_PMON_1
24	X	SPEEDO_PMON_0
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	HSI

Bit	Reset	Description
14	X	UARTC
13	X	ACTMON
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	DTV
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.3.7 SEC_ICTLR_FIR_SET_0

Force Interrupt Register Set

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

Offset: 118h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	SPEEDO_PMON_1
24	0x0	SPEEDO_PMON_0
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EVENT_GPIO_D

Bit	Reset	Description
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	DTV
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.3.8 SEC_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

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

Offset: 11ch | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	SPEEDO_PMON_1
24	0x0	SPEEDO_PMON_0
23	0x0	GPIO5

Bit	Reset	Description
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	DTV
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.3.9 SEC_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

IER31_IER0: Interrupt Enable Status. 0 = Disabled.

Offset: 120h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1

Bit	Reset	Description
26	X	SE
25	X	SPEEDO_PMON_
24	X	SPEEDO_PMON_0
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU
10	X	TMR4
9	X	TMR3
8	X	TWC
7	X	DTV
6	X	I2C
5	X	UARTB
4	X	UARTA
3	X	GPIO4
2	X	GPIO3
1	X	GPIO2
0	X	GPIO1

3.2.3.10 SEC_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

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

Offset: 124h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	DMA_RX

Bit	Reset	Description
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	SPEEDO_PMON_1
24	0x0	SPEEDO_PMON_0
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	DTV
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.3.11 SEC_ICTLR_CPU_IER_CLR_0

CPUs Clear Interrupt Enable for CPU

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

Offset: 128h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	SPEEDO_PMON_1
24	0x0	SPEEDO_PMON_0
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	DTV
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4

Bit	Reset	Description
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.3.12 SEC_ICTLR_CPU_IEP_CLASS_0

CPUs Interrupt Enable Priority Class (FIQ/IRQ)

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

Offset: 12ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	SPEEDO_PMON_1
24	0x0	SPEEDO_PMON_0
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	DTV

Bit	Reset	Description
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.3.13 SEC_ICTLR_COP_IER_0

Enabled Interrupt Source for COP Register

IER31_IER0: Interrupt Enable Status. 0 = Disabled.

Offset: 130h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	DMA_RX
30	X	DMA_TX
29	X	AHB_DMA_COP
28	X	APB_DMA_COP
27	X	SBC1
26	X	SE
25	X	SPEEDO_PMON_1
24	X	SPEEDO_PMON_0
23	X	GPIO5
22	X	STAT_MON
21	X	I2C5
20	X	VFIR
19	X	EVENT_GPIO_D
18	X	EVENT_GPIO_C
17	X	EVENT_GPIO_B
16	X	EVENT_GPIO_A
15	X	HSI
14	X	UARTC
13	X	ACTMON
12	X	FLOW_RSM_COP
11	X	FLOW_RSM_CPU

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

3.2.3.14 SEC_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

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

Offset: 134h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	SPEEDO_PMON_1
24	0x0	SPEEDO_PMON_0
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	HSI

Bit	Reset	Description
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	DTV
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.3.15 SEC_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

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

Offset: 138h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	SPEEDO_PMON_1
24	0x0	SPEEDO_PMON_0
23	0x0	GPIO5
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EVENT_GPIO_D

Bit	Reset	Description
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	DTV
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.3.16 SEC_ICTLR_COP_IEP_CLASS_0

COPs Interrupt Enable Priority Class (FIQ/IRQ) Register

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

Offset: 13ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	DMA_RX
30	0x0	DMA_TX
29	0x0	AHB_DMA_COP
28	0x0	APB_DMA_COP
27	0x0	SBC1
26	0x0	SE
25	0x0	SPEEDO_PMON_1
24	0x0	SPEEDO_PMON_0
23	0x0	GPIO5

Bit	Reset	Description
22	0x0	STAT_MON
21	0x0	I2C5
20	0x0	VFIR
19	0x0	EVENT_GPIO_D
18	0x0	EVENT_GPIO_C
17	0x0	EVENT_GPIO_B
16	0x0	EVENT_GPIO_A
15	0x0	HSI
14	0x0	UARTC
13	0x0	ACTMON
12	0x0	FLOW_RSM_COP
11	0x0	FLOW_RSM_CPU
10	0x0	TMR4
9	0x0	TMR3
8	0x0	TWC
7	0x0	DTV
6	0x0	I2C
5	0x0	UARTB
4	0x0	UARTA
3	0x0	GPIO4
2	0x0	GPIO3
1	0x0	GPIO2
0	0x0	GPIO1

3.2.3.17 SEC_ICTLR_DRQ_TX_STATUS_0

Second Interrupt Controller DRQ TX Registers.

DRQ Interrupt Source Status

DRQ31_DRQ0: DRQ_INTERRUPT_SOURCE_STATUS

Offset: 140h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
19	X	I2C5_TX
18	X	I2C4_TX
17	X	VFIR_TX_FIFO
16	X	I2C3_TX

Bit	Reset	Description
15	X	UARTE_OUTPUT_FIFO
14	X	UARTD_OUTPUT_FIFO
13	X	I2C2_TX
12	X	I2C_TX
11	X	TWC_TX
10	X	UARTC_OUTPUT_FIFO
9	X	UARTB_OUTPUT_FIFO
8	X	UARTA_OUTPUT_FIFO
7	X	SPI_TX
4	X	APBDMA_TX_4
3	X	APBDMA_TX_3
2	X	APBDMA_TX_2
1	X	APBDMA_TX_1

3.2.3.18 SEC_ICTLR_DRQ_TX_ENABLE_0

DRQ Interrupt Source Enable Gates

DER31_DER0: DRQ_INTERRUPT_SOURCE_ENABLE_GATES

Offset: 144h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	R/W	Reset	Description
19	RO	X	I2C5_TX
18	RO	X	I2C4_TX
17	RO	X	VFIR_TX_FIFO
16	RO	X	I2C3_TX
15	RO	X	UARTE_OUTPUT_FIFO
14	RO	X	UARTD_OUTPUT_FIFO
13	RO	X	I2C2_TX
12	RO	X	I2C_TX
11	RO	X	TWC_TX
10	RO	X	UARTC_OUTPUT_FIFO
9	RO	X	UARTB_OUTPUT_FIFO
8	RO	X	UARTA_OUTPUT_FIFO
7	RO	X	SPI_TX
4	RO	X	APBDMA_TX_4
3	RO	X	APBDMA_TX_3

Bit	R/W	Reset	Description
2	RO	X	APBDMA_TX_2
1	RO	X	APBDMA_TX_1

3.2.3.19 SEC_ICTLR_DRQ_RX_STATUS_0

Second interrupt controller DRQ RX registers.

DRQ Interrupt Source Status por=0x00000000

DRQ31_DRQ0: DRQ_INTERRUPT_SOURCE_STATUS

Offset: 148h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
19	X	I2C5_RX
18	X	I2C4_RX
17	X	VFIR_RX_FIFO
16	X	I2C3_RX
15	X	UARTE_INPUT_FIFO
14	X	UARTD_INPUT_FIFO
13	X	I2C2_RX
12	X	I2C_RX
11	X	TWC_RX
10	X	UARTC_INPUT_FIFO
9	X	UARTB_INPUT_FIFO
8	X	UARTA_INPUT_FIFO
7	X	SPI_RX
4	X	APBDMA_RX_4
3	X	APBDMA_RX_3
2	X	APBDMA_RX_2
1	X	APBDMA_RX_1

3.2.3.20 SEC_ICTLR_DRQ_RX_ENABLE_0

DRQ Interrupt Source Enable Gates

DER31_DER0: DRQ_INTERRUPT_SOURCE_ENABLE_GATES

Offset: 14ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	R/W	Reset	Description
19	RO	X	I2C5_RX
18	RO	X	I2C4_RX
17	RO	X	VFIR_RX_FIFO
16	RO	X	I2C3_RX
15	RO	X	UARTE_INPUT_FIFO
14	RO	X	UARTD_INPUT_FIFO
13	RO	X	I2C2_RX
12	RO	X	I2C_RX
11	RO	X	TWC_RX
10	RO	X	UARTC_INPUT_FIFO
9	RO	X	UARTB_INPUT_FIFO
8	RO	X	UARTA_INPUT_FIFO
7	RO	X	SPI_RX
4	RO	X	APBDMA_RX_4
3	RO	X	APBDMA_RX_3
2	RO	X	APBDMA_RX_2
1	RO	X	APBDMA_RX_1

3.2.4 Third Interrupt Controller Registers

Third interrupt controller base registers.

3.2.4.1 TRI_ICTLR_VIRQ_CPU_0

Valid Interrupt Request Status for CPU Register

IRQ31_IRQ0: Flags set by Hardware, cleared by Software.

Offset: 200h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

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

3.2.4.2 TRI_ICTLR_VIRQ_COP_0

Valid Interrupt Status for COP Register

IRQ31_IRQ0: Flags set by Hardware, cleared by Software.

Offset: 204h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
27	X	UARTE
26	X	UARTD
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
16	X	NOR_FLASH
15	X	SBC6
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU

Bit	Reset	Description
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.4.3 TRI_ICTLR_VFIQ_CPU_0

FIQ Valid Interrupt Status for CPU Register

FIQ31_FIQ0: Flags set by Hardware, cleared by Software.

Offset: 208h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
27	X	UARTE
26	X	UARTD
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
16	X	NOR_FLASH
15	X	SBC6
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP

Bit	Reset	Description
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.4.4 TRI_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for COP Register

FIQ31_FIQ0: Flags set by Hardware, cleared by Software.

Offset: 20ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
27	X	UARTE
26	X	UARTD
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
16	X	NOR_FLASH
15	X	SBC6
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI

Bit	Reset	Description
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.4.5 TRI_ICTLR_ISR_0

Latched Interrupt Status Register (HW)

ISR31_ISR0: Set by hardware event, cleared at source by software.

Offset: 210h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
27	X	UARTE
26	X	UARTD
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
16	X	NOR_FLASH
15	X	SBC6

Bit	Reset	Description
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.4.6 TRI_ICTLR_FIR_0

Forced Interrupt Status Register (SW)

FIR31_FIR0: Set during write to FIR_SET, cleared during write to FIR_CLR.

Offset: 214h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
27	X	UARTE
26	X	UARTD
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3

Bit	Reset	Description
18	X	SBC2
17	X	HDA
16	X	NOR_FLASH
15	X	SBC6
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.4.7 TRI_ICTLR_FIR_SET_0

Force Interrupt Register Set

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

Offset: 218h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UARTE
26	0x0	UARTD
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6

Bit	Reset	Description
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
16	0x0	NOR_FLASH
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.4.8 TRI_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

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

Offset: 21ch | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UARTE

Bit	Reset	Description
26	0x0	UARTD
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
16	0x0	NOR_FLASH
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.4.9 TRI_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

IER31_IER0: Interrupt Enable Status. 0 = Disabled.

Offset: 220h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
27	X	UARTE
26	X	UARTD
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2
19	X	SBC3
18	X	SBC2
17	X	HDA
16	X	NOR_FLASH
15	X	SBC6
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU

Bit	Reset	Description
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.4.10 TRI_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

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

Offset: 224h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UARTE
26	0x0	UARTD
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
16	0x0	NOR_FLASH
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP

Bit	Reset	Description
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.4.11 TRI_ICTLR_CPU_IER_CLR_0

CPUs Clear Interrupt Enable for CPU

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

Offset: 228h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UARTE
26	0x0	UARTD
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
16	0x0	NOR_FLASH
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI

Bit	Reset	Description
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.4.12 TRI_ICTLR_CPU_IEP_CLASS_0

CPUs Interrupt Enable Priority Class (FIQ/IRQ)

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

Offset: 22ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UARTE
26	0x0	UARTD
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
16	0x0	NOR_FLASH
15	0x0	SBC6

Bit	Reset	Description
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
31:0	0x0	CPU_IEP_CLASS: Set Priority Interrupt Source For CPU. 1 = FIQ, 0 = IRQ.
0	0x0	HOST1X_SYNCPT_COP

3.2.4.13 TRI_ICTLR_COP_IER_0

Enabled Interrupt Source for COP Register

IER31_IER0: Interrupt Enable Status. 0 = Disabled.

Offset: 230h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SW_INTR
30	X	SBC5
29	X	SBC4
28	X	I2C3
27	X	UARTE
26	X	UARTD
25	X	GPIO7
24	X	TVDAC
23	X	GPIO6
22	X	PMU_EXT
21	X	KBC
20	X	I2C2

Bit	Reset	Description
19	X	SBC3
18	X	SBC2
17	X	HDA
16	X	NOR_FLASH
15	X	SBC6
14	X	EMC
13	X	MC
12	X	TVO
11	X	HDMI
10	X	DISPLAYB
9	X	DISPLAY
8	X	GR2D
7	X	ISP
6	X	EPP
5	X	VI
4	X	MPE
3	X	HOST1X_GEN_CPU
2	X	HOST1X_GEN_COP
1	X	HOST1X_SYNCPT_CPU
0	X	HOST1X_SYNCPT_COP

3.2.4.14 TRI_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

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

Offset: 234h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UARTE
26	0x0	UARTD
25	0x0	GPIO7
24	0x0	TVDAC

Bit	Reset	Description
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
16	0x0	NOR_FLASH
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.4.15 TRI_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

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

Offset: 238h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3

Bit	Reset	Description
27	0x0	UARTE
26	0x0	UARTD
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
16	0x0	NOR_FLASH
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.4.16 TRI_ICTLR_COP_IEP_CLASS_0

COPs Interrupt Enable Priority Class (FIQ/IRQ) Register

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

Offset: 23ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	SW_INTR
30	0x0	SBC5
29	0x0	SBC4
28	0x0	I2C3
27	0x0	UARTE
26	0x0	UARTD
25	0x0	GPIO7
24	0x0	TVDAC
23	0x0	GPIO6
22	0x0	PMU_EXT
21	0x0	KBC
20	0x0	I2C2
19	0x0	SBC3
18	0x0	SBC2
17	0x0	HDA
16	0x0	NOR_FLASH
15	0x0	SBC6
14	0x0	EMC
13	0x0	MC
12	0x0	TVO
11	0x0	HDMI
10	0x0	DISPLAYB
9	0x0	DISPLAY
8	0x0	GR2D
7	0x0	ISP
6	0x0	EPP
5	0x0	VI
4	0x0	MPE
3	0x0	HOST1X_GEN_CPU

Bit	Reset	Description
2	0x0	HOST1X_GEN_COP
1	0x0	HOST1X_SYNCPT_CPU
0	0x0	HOST1X_SYNCPT_COP

3.2.5 Fourth Interrupt Controller Registers

Fourth interrupt controller base registers

3.2.5.1 QUAD_ICTLR_VIRQ_CPU_0

Valid Interrupt Request Status for CPU Register

IRQ31_IRQ0: Flags set by Hardware, cleared by Software.

Offset: 300h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	TMR_SHARED
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

Bit	Reset	Description
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
6	X	TSENSOR
5	X	AVP_CACHE
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
0	X	SNOR

3.2.5.2 QUAD_ICTLR_VIRQ_COP_0

Valid Interrupt Status for COP Register

IRQ31_IRQ0: Flags set by Hardware, cleared by Software.

Offset: 304h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	TMR_SHARED
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

Bit	Reset	Description
11	X	APB_DMA_CH3
10	X	APB_DMA_CH2
9	X	APB_DMA_CH1
8	X	APB_DMA_CH0
7	X	AUDIO_CLUSTER
6	X	TSENSOR
5	X	AVP_CACHE
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
0	X	SNOR

3.2.5.3 QUAD_ICTLR_VFIQ_CPU_0

FIQ Valid Interrupt Status for CPU Register

FIQ31_FIQ0: Flags set by Hardware, cleared by Software.

Offset: 308h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	TMR_SHARED
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

Bit	Reset	Description
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
6	X	TSENSOR
5	X	AVP_CACHE
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
0	X	SNOR

3.2.5.4 QUAD_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for COP Register

FIQ31_FIQ0: Flags set by Hardware, cleared by Software.

Offset: 30ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	TMR_SHARED
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

Bit	Reset	Description
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
6	X	TSENSOR
5	X	AVP_CACHE
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
0	X	SNOR

3.2.5.5 QUAD_ICTLR_ISR_0

Latched Interrupt Status Register (HW)

ISR31_ISR0: Set by hardware event, cleared at source by software.

Offset: 310h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	TMR_SHARED
25	X	TMR5
24	X	I2C4
23	X	APB_DMA_CH15
22	X	APB_DMA_CH14
21	X	APB_DMA_CH13

Bit	Reset	Description
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
6	X	TSENSOR
5	X	AVP_CACHE
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
0	X	SNOR

3.2.5.6 QUAD_ICTLR_FIR_0

Forced Interrupt Status Register (SW)

FIR31_FIR0: Set during write to FIR_SET, cleared during write to FIR_CLR.

Offset: 314h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	TMR_SHARED
25	X	TMR5
24	X	I2C4

Bit	Reset	Description
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
6	X	TSENSOR
5	X	AVP_CACHE
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
0	X	SNOR

3.2.5.7 QUAD_ICTLR_FIR_SET_0

Force Interrupt Register Set

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

Offset: 318h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU

Bit	Reset	Description
26	0x0	TMR_SHARED
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
6	0x0	TSENSOR
5	0x0	AVP_CACHE
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
0	0x0	SNOR

3.2.5.8 QUAD_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

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

Offset: 31ch | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	TMR_SHARED
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
6	0x0	TSENSOR
5	0x0	AVP_CACHE
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT

Bit	Reset	Description
1	0x0	USB3
0	0x0	SNOR

3.2.5.9 QUAD_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

IER31_IER0: Interrupt Enable Status. 0 = Disabled.

Offset: 320h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	TMR_SHARED
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
6	X	TSENSOR
5	X	AVP_CACHE

Bit	Reset	Description
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
0	X	SNOR

3.2.5.10 QUAD_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

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

Offset: 324h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	TMR_SHARED
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

Bit	Reset	Description
7	0x0	AUDIO_CLUSTER
6	0x0	TSENSOR
5	0x0	AVP_CACHE
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
0	0x0	SNOR

3.2.5.11 QUAD_ICTLR_CPU_IER_CLR_0

CPUs Clear Interrupt Enable for CPU

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

Offset: 328h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	TMR_SHARED
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

Bit	Reset	Description
10	0x0	APB_DMA_CH2
9	0x0	APB_DMA_CH1
8	0x0	APB_DMA_CH0
7	0x0	AUDIO_CLUSTER
6	0x0	TSENSOR
5	0x0	AVP_CACHE
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
0	0x0	SNOR

3.2.5.12 QUAD_ICTLR_CPU_IEP_CLASS_0

CPUs Interrupt Enable Priority Class (FIQ/IRQ)

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

Offset: 32ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	TMR_SHARED
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

Bit	Reset	Description
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
6	0x0	TSENSOR
5	0x0	AVP_CACHE
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
0	0x0	SNOR

3.2.5.13 QUAD_ICTLR_COP_IER_0

Enabled Interrupt Source for COP Register

IER31_IER0: Interrupt Enable Status. 0 = Disabled.

Offset: 330h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	CAR
29	X	GPIO8
28	X	WDT_AVP
27	X	WDT_CPU
26	X	TMR_SHARED
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

Bit	Reset	Description
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
6	X	TSENSOR
5	X	AVP_CACHE
4	X	PCIE_WAKE
3	X	PCIE_MSI
2	X	PCIE_INT
1	X	USB3
0	X	SNOR

3.2.5.14 QUAD_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

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

Offset: 334h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	TMR_SHARED
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

Bit	Reset	Description
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
6	0x0	TSENSOR
5	0x0	AVP_CACHE
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
0	0x0	SNOR

3.2.5.15 QUAD_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

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

Offset: 338h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	TMR_SHARED
25	0x0	TMR5
24	0x0	I2C4
23	0x0	APB_DMA_CH15

Bit	Reset	Description
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
6	0x0	TSENSOR
5	0x0	AVP_CACHE
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
0	0x0	SNOR

3.2.5.16 QUAD_ICTLR_COP_IEP_CLASS_0

COPs Interrupt Enable Priority Class (FIQ/IRQ) Register

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

Offset: 33ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
30	0x0	CAR
29	0x0	GPIO8
28	0x0	WDT_AVP
27	0x0	WDT_CPU
26	0x0	TMR_SHARED

Bit	Reset	Description
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
6	0x0	TSENSOR
5	0x0	AVP_CACHE
4	0x0	PCIE_WAKE
3	0x0	PCIE_MSI
2	0x0	PCIE_INT
1	0x0	USB3
0	0x0	SNOR

3.2.6 Fifth Interrupt Controller Registers

Fifth interrupt controller base registers

3.2.6.1 PENTA_ICTLR_VIRQ_CPU_0

Valid Interrupt Request Status for CPU Register

IRQ31_IRQ0: Flags set by Hardware, cleared by Software.

Offset: 400h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
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

3.2.6.2 PENTA_ICTLR_VIRQ_COP_0

Valid Interrupt Status for COP Register

IRQ31_IRQ0: Flags set by Hardware, cleared by Software.

Offset: 404h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
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

3.2.6.3 PENTA_ICTLR_VFIQ_CPU_0

FIQ Valid Interrupt Status for CPU Register

FIQ31_FIQ0: Flags set by Hardware, cleared by Software.

Offset: 408h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
19	X	CPU3_PMU_INTR
18	X	CPU2_PMU_INTR
17	X	CPU1_PMU_INTR
16	X	CPU0_PMU_INTR

Bit	Reset	Description
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

3.2.6.4 PENTA_ICTLR_VFIQ_COP_0

FIQ Valid Interrupt Status for COP Register

FIQ31_FIQ0: Flags set by Hardware, cleared by Software.

Offset: 40ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
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

Bit	Reset	Description
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

3.2.6.5 PENTA_ICTLR_ISR_0

Latched Interrupt Status Register (HW)

ISR31_ISR0: Read-only. Set by hardware event, cleared at source by software.

Offset: 410h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
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

3.2.6.6 PENTA_ICTLR_FIR_0

Forced Interrupt Status Register (SW)

FIR31_FIR0: Read only: Set during write to FIR_SET, cleared during write to FIR_CLR.

Offset: 414h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
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

3.2.6.7 PENTA_ICTLR_FIR_SET_0

Force Interrupt Register Set

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

Offset: 418h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR

Bit	Reset	Description
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

3.2.6.8 PENTA_ICTLR_FIR_CLR_0

Force Interrupt Register Clear Register

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

Offset: 41ch | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
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

Bit	Reset	Description
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

3.2.6.9 PENTA_ICTLR_CPU_IER_0

Enabled Interrupt Source for CPU Register

IER31_IER0: Interrupt Enable Status. 0 = Disabled.

Offset: 420h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
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

3.2.6.10 PENTA_ICTLR_CPU_IER_SET_0

Set Interrupt Enable for CPU Register

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

Offset: 424h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
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

3.2.6.11 PENTA_ICTLR_CPU_IER_CLR_0

CPUs Clear Interrupt Enable for CPU

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

Offset: 428h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR

Bit	Reset	Description
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

3.2.6.12 PENTA_ICTLR_CPU_IEP_CLASS_0

CPUs Interrupt Enable Priority Class (FIQ/IRQ)

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

Offset: 42ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
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

Bit	Reset	Description
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

3.2.6.13 PENTA_ICTLR_COP_IER_0

Enabled Interrupt Source for COP Register

IER31_IER0: Interrupt Enable Status. 0 = Disabled.

Offset: 430h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
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

3.2.6.14 PENTA_ICTLR_COP_IER_SET_0

Set Interrupt Source for COP Register

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

Offset: 434h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
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

3.2.6.15 PENTA_ICTLR_COP_IER_CLR_0

Clear Interrupt Source for COP Register

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

Offset: 438h | Read/Write: WO | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
19	0x0	CPU3_PMU_INTR
18	0x0	CPU2_PMU_INTR
17	0x0	CPU1_PMU_INTR
16	0x0	CPU0_PMU_INTR

Bit	Reset	Description
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

3.2.6.16 PENTA_ICTLR_COP_IEP_CLASS_0

COPs Interrupt Enable Priority Class (FIQ/IRQ) Register

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

Offset: 43ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
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

Bit	Reset	Description
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

[THIS PAGE INTENTIONALLY LEFT BLANK]

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. An alternative mechanism has been added in Tegra® 3 devices called the Atomic block. Please refer to the corresponding section of this document for further details.

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).

4.2 Semaphore Registers

4.2.1 ARB_SEMA_SMP_GNT_ST_0

Semaphore Granted Status Register

Offset: 000h | Read/Write: RO | Reset: 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: 004h | Read/Write: WO | Reset: 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: 008h | Read/Write: R/W | Reset: 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 (1=PENDING) Register

Offset: 00ch | Read/Write: RO | Reset: 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

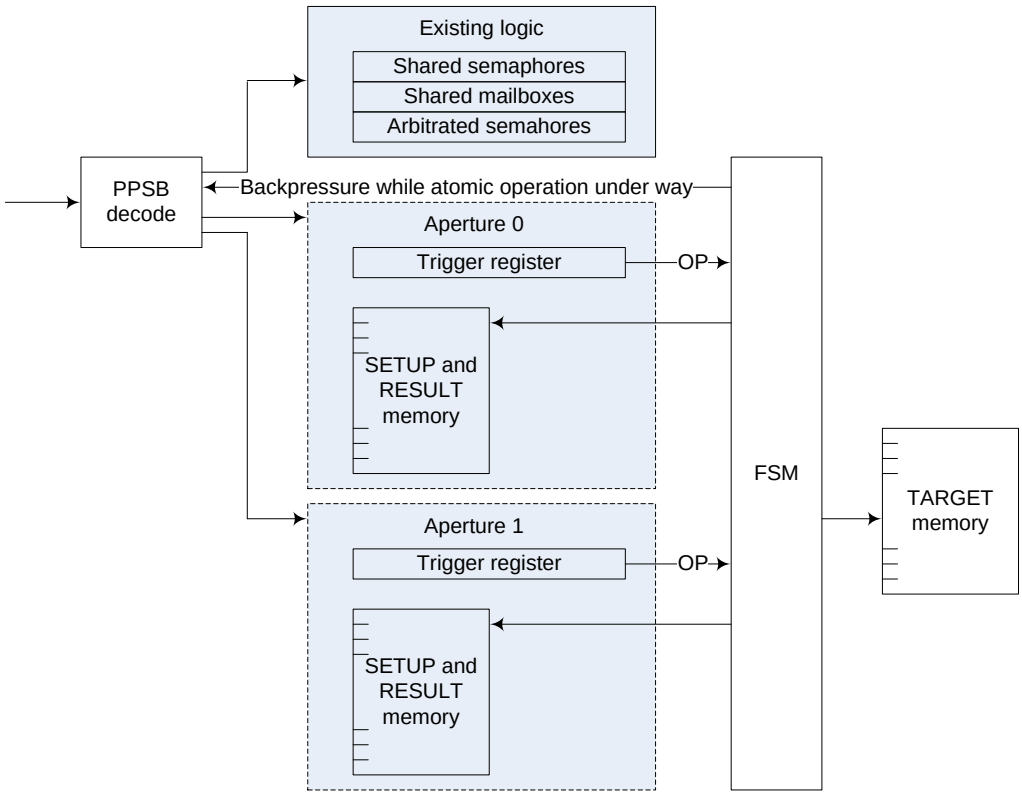
5.1 Introduction

The Atomics block is a new block added to Tegra® 3 to assist software tasks with maintaining synchronization. It is parallel to the existing Semaphores block, and intended to extend that block's functionality with new capabilities.

5.2 Functionality

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

Figure 2. 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 3, normally one used by CPU, the other by AVP, but this is not enforced by HW as it is for the arbitrated semaphores.

Setup, target and result registers are 32-bits each with no further structure imposed by HW. 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 HW. Typically, the HW implementation has no intermediate values if operations are performed in one clock cycle.
- Arithmetic operations are performed as signed 32 bits operations modulo 2^{32} using 2s 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.

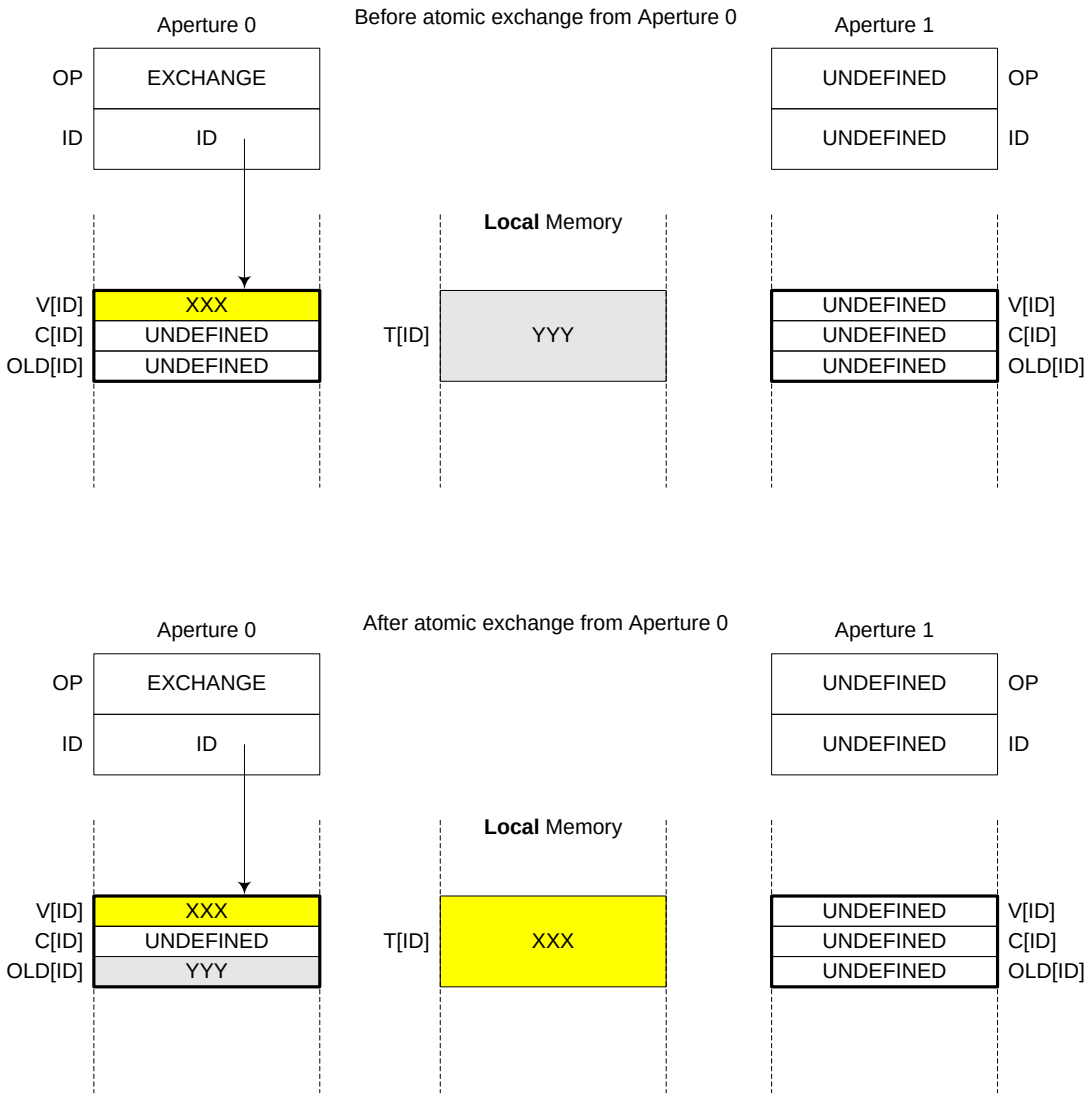
The corresponding pseudo code is

```
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 3. 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 HW 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 HW 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

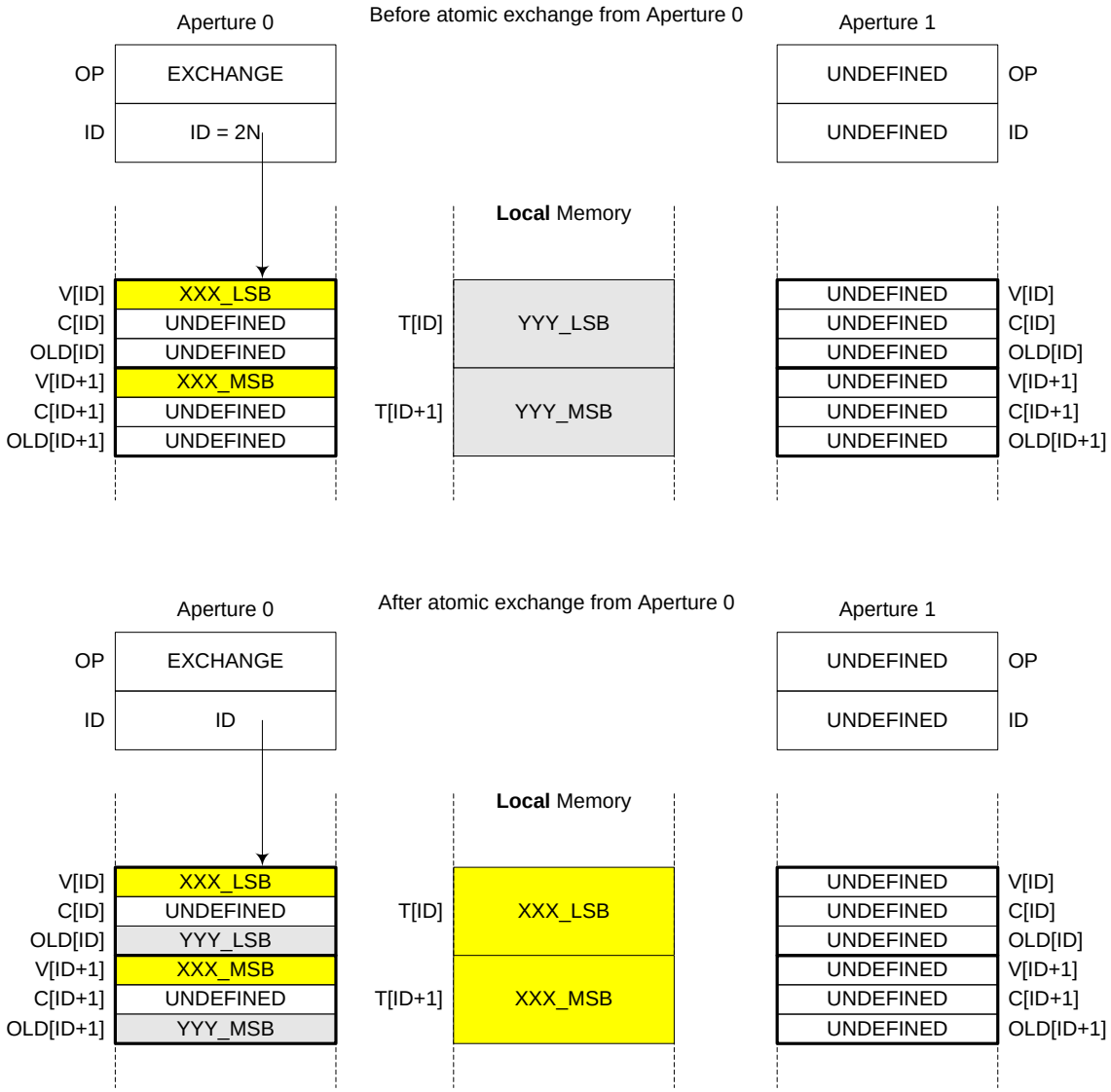
These are simple extensions of previous cases, where the operation is either a bit set, bit clear or bit invert.

```
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 bits operations use an even/odd pair of address, with the MSB at the odd address. The next figure shows how a 64 bits 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 4. 64 bit atomic exchange operation



5.4 Atomics Registers

5.4.1 ATOMICS_AP0_TRIGGER_0

Trigger Registers

Offset: 000h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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: 004h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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: 400h..5ffh | Read/Write: R/W | Reset: 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: 800h..9ffh | Read/Write: R/W | Reset: 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: c00h..dffh | Read/Write: RO | Reset: 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: 1000h..11ffh | Read/Write: R/W | Reset: 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: 1400h..15ffh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	V

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: 1800h..19ffh | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	R

[THIS PAGE INTENTIONALLY LEFT BLANK]

6.0 CLOCK AND RESET CONTROLLER

Note: PRELIMINARY pending review

The Clock and Reset (CAR) block comprises CLKGEN and RSTGEN.

CLKGEN provides the registers to program the PLLs and controls most of the clock source programming, and most of the clock dividers. CLKGEN input signals include the external clock for the reference frequency (12 MHz, 26 MHz,) and the external clock for the Real Time Clock (32.768 KHz). The outputs from CLKGEN are mostly clocks to the other blocks in the Tegra® 3 system.

RSTGEN provides the registers needed to control resetting each block in the Tegra 3 system.

6.1 Hardware Features

6.1.1 External Clock Sources

The CLKGEN block takes two external clock sources as input:

- A single external 32.768 KHz clock, normally provided by the Power Management Unit (PMU)
- A single crystal oscillator at either 12 MHz or 26 MHz. 12 MHz should be used when the Tegra 3 device has a dedicated crystal. Support for the 26 MHz rate may be useful for cell phone applications, where it is expected to be provided by the Baseband Processor. Lower frequencies are generally lower power.

6.1.2 PLLs

There are 12 PLLs in the Tegra 3 device's clock system:

- PLLX is dedicated to the CPU complex, and provides a high speed CPU clock when required. It operates up to 1.1 GHz or higher.
- PLLC is a general purpose PLL, whose nominal maximum frequency is 600 MHz. This PLL is available as a clock source to many modules.
- PLLP is a "fixed" 408 MHz source for most of the Peripherals on the Tegra 3 devices. It should always be set to this frequency. It is available as a source to most modules. In addition, PLLP is the source of a number of standard divided-down frequencies that are also available as clock sources to other modules.
- PLLM is the source for the External Memory Controller (EMC) 2x clock. It is dedicated to this function. The nominal maximum frequency required by EMC on the Tegra 3 devices is 800 MHz. PLLM is available as a source to other modules as well.
- PLLA is the source for Audio, and is used to generate exact 11.2896, 12.288 or 24.576 MHz for audio codecs and audio-related devices. When the audio clock is provided by a PLL in the external audio codec, PLLA is available as a general purpose PLL.
- PLLD and PLLD2 are the sources for DSI and for the Display subsystem in general. They provide the frequencies required by DSI and the Display controller when DSI is in use. They can also be used as dedicated Display PLLs to provide the frequencies required by Display, HDMI, TVOUT when a precise enough frequency cannot be provided by one of the other Tegra 3 device PLLs. Wherever possible, the Display subsystem should use a clock source other than PLLD and PLLD2, since these PLLs consumes higher power than the other (non-MIPI) PLLs in the system.
- PLLU is the source for the USB PHY and controllers. It is dedicated to provide the 12 MHz required by the USB PHY, and 60 MHz and 480 MHz USB clocks.

- refPLLe and PLLE work together to generate the required 100 MHz reference clock used for the USB3.0 and is capable of generating a spread-spectrum clock.
- Two additional PLLs are embedded in their respective interface controllers. The USB controller has an internal PLL as does the TMDS HDMI PHY. These PLLs are programmed via their respective controllers' registers, and the clocks generated by these PLLs are neither visible to CLKGEN nor to the rest of the system.

6.1.3 Clock Dividers / Skippers / Multipliers

There are a number of clock skippers, clock dividers, and clock multipliers in use in the Tegra 3 devices.

- An M/N clock-skipping divider that is used to generate the CPU clock and also the Tegra 3 "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).
- An unsigned 16-bit divider (U16) that is the divider for the I2C and UART devices.
- A times-two multiplier used to double the external reference frequency, and used to double the audio clock.
- An unsigned 8b divider that provides 7b of mantissa and 1b of fraction (U7.1). Tegra 3 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 3 devices.

Note that 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.

6.1.4 Clock Sources

The clock sources are described as follows:

- PLLM: Clock source for EMC 2x clock
- PLLX: Clock source for the CPU
- PLLC: Clock source for general use
- 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 display subsystem
- OSC: Source from external crystal
- CLK_32K_IN: 32-KHz clock provided by the PMC

Almost every device or block in the Tegra 3 can be driven from a selection of clock sources. There are three general types of source selection:

- Single source to module (with or without a divider)
- 4-way MUX of clock sources to module (with or without divider)
- 8-way MUX of clock sources to module.

Inputs to clock MUXes are identified as primary (1o), secondary (2o), tertiary (3o), etc. The Power-On-Reset values of all MUXes always select the “osc” clock source. For those MUXes which has no “osc” as clock source, the Power-On-Reset values always select the primary clock source. The clocks available as sources in the Tegra 3 devices are shown in Table 9.

Table 9. Primary Clock Sources in the Tegra 3 Devices

Clock name	Description
dbg_oscout	This clock runs at 12 MHz or 26 MHz.
car_clk_m	This clock (with DFT control) runs at 12 MHz or 26 MHz.
ck32khz_IB	This clock runs at 32.768 KHz.
car_clk_s	This clock (with DFT control) runs at 32.768 KHz.
car_sclk	This is the system clock which can run at max 200 MHz.
pllP_out	This is the PLLP's output clock.
pllC_out	This is the PLLC's output clock.
pllM_out	This is the PLLM's output clock.
int_pllA_out	This is the PLLA's output clock.
pllD_out0	This is the divided-by-2 (max @ 500 MHz) from PLLD's output clock.
pllX_out	This is the PLLX's output clock (only use in mini-CAR CPU)

Table 10. Derived Clock Sources in the Tegra 3 Devices

Clock name	Description
clk_d:	This is the clock doubled from "dbg_oscout"
pllC_out1:	This is the NV divider output clocked by "pllC_out".
pllP_out1:	This is the NV divider output (fixed @ 9.6 MHz) clocked by "pllP_out".
pllP_out2:	This is the NV divider output (fixed @ 48 MHz) clocked by "pllP_out".
pllP_out3:	This is the NV divider output (fixed @ 102 MHz) clocked by "pllP_out".
pllP_out4:	This is the NV divider output (fixed @ (102 MHz) clocked by "pllP_out".
pllM_out1:	This is the NV divider output clocked by "pllM_out".
pllA_out0:	This is the NV divider output clocked by "int_pllA_out".
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".

Table 11 Clock Source Muxes Summary

	dbg_oscout	pllC_out	ck32khz_IB	pllM_out	pllM_2x_out	pllP_out	pllP_out4	pllP_out3	clk_d	pllX_out	pllC_out1	pllP_out2	pllM_out1	pllA_out0	MOD_audio_2x_sync_clk	pllD_out0	pllE_out0	Have super clock divider ? (# of bits)	Have NV divider? (# of bits)
CPU	0	1	2	3		4	5	6	7	8								8	
CPU_LP	0,*8	1	2	3		4	5	6	7									8	
SYS/COP	0		6				2	3	5		1	4	7					8	
SPDIF_OUT	3					2								0	1				8
SPDIF_IN		1		2		0													8
PWM	2	1	3			0													8
DTV	3	1		2		0													8
SBC[1:6]	3	1		2		0													8
NAND	3	1		2		0													8
VFIR	3	1		2		0													8
SDMMC[1:4]	3	1		2		0													8
VDE	3	1		2		0													8
CSITE	3	1		2		0													8
LA	3	1		2		0													8
OWR	3	1		2		0													8
NOR	3	1		2		0													8
MIPI	3	1		2		0													8
I2C[1:4]	3	1		2		0													16
DVC	3	1		2		0													16
UART[A:E]	3	1		2		0													16
3D		1		0		2								3					8
3D2		1		0		2								3					8
2D		1		0		2								3					8
VI		1		0		2								3					8
VI_SENSOR		1		0		2								3					8
EPP		1		0		2								3					8
MPE		1		0		2								3					8
HOST1X		1		0		2								3					8
CVE	3	2				0										1			8
TVO	3	2				0										1			8
HDMI	3	2				0										1			8
TVDAC	3	2				0										1			8
DISP[1:2]	3	2				0										1			8
EMC (2x/1x)	3	1		0	4a	2													8
AUDIO	3	1				2								0					8
APBIF	3	1				2								0					8
DAM[0:2]	3	1				2								0					8
I2S[0:4]	3					2								0	1				8
HDA	3	1				2								0					8

	dbg_oscout	pllC_out	ck32khz_IB	pllM_out	pllM_2x_out	pllP_out	pllP_out4	pllP_out3	clk_d	pllX_out	pllC_out1	pllP_out2	pllM_out1	pllA_out0	MOD_audio_2x_sync_clk	pllD_out0	pllE_out0	Have super clock divider ? (# of bits)	Have NV divider? (# of bits)
HDA2CODEC_2X	3	1				0	2												8
SE	3	1		2		0													8
SATA	3	1		2		0													8
SATA_OOB	3	1		2		0													8
SATA_FPCI	3	1		2		0													8
MSEL	3	1		2		0													8
TSENSOR	2	1	3			0													8
ACTMON	2	1	3			0													8
EXTPERIPH[1:3]	3		1			2								0			4		8

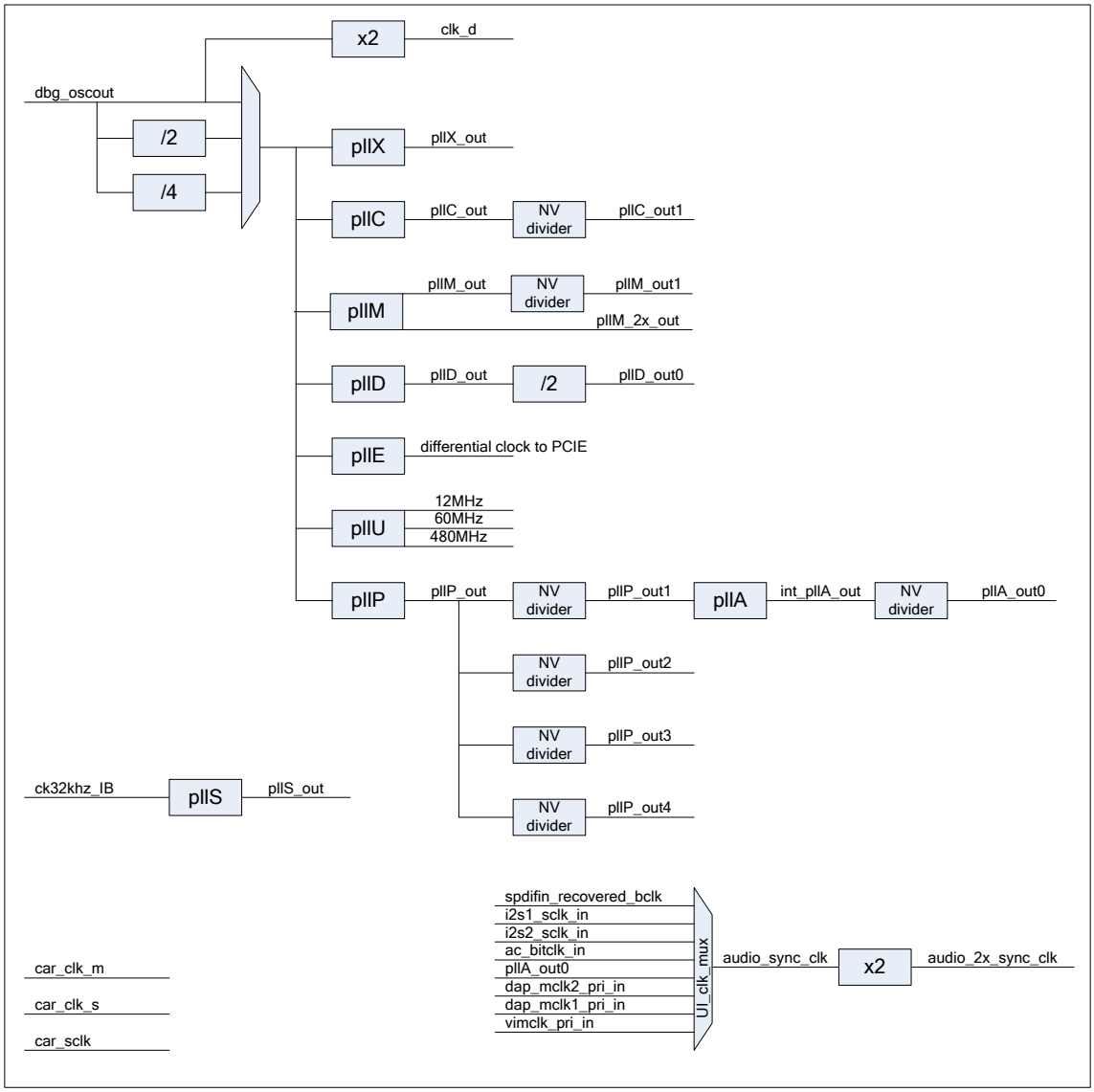
Note: [a] When EMC clock source = 4, the NV divider will be bypassed/ignored. This setting will give a very short low jitter clock path from pllM_out to EMC clock.
[b] Here, max frequency of 800 MHz is the input to the NV divider, not the divider output. It is listed here for DFT testing purposes only.

The above table provides a quick view of the clock related information for each device. For example, CSITE has 4 clock sources (00=pllP_out, 01=pllC_out, 10=pllM_out, and 11=osc) and has an 8-bit wide NV divider.

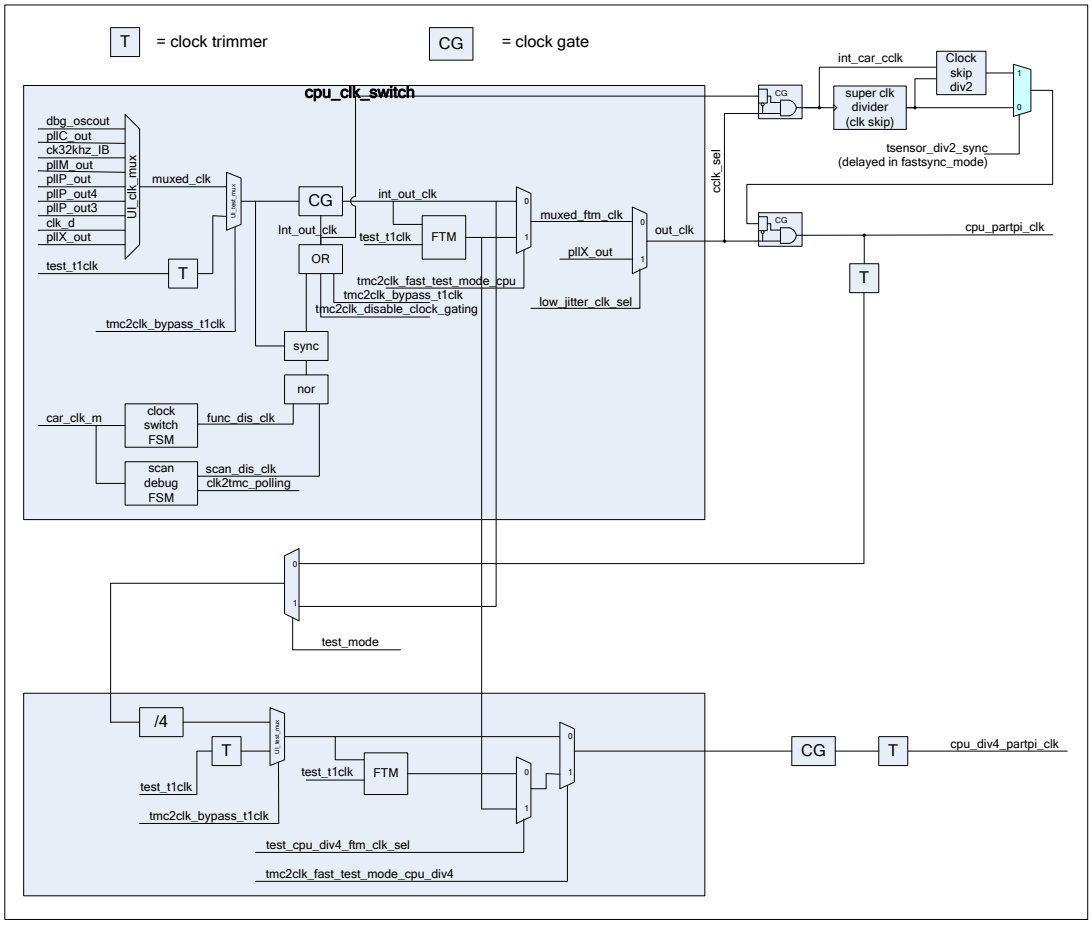
	dbg_oscout	pllC_out	ck32khz_IB	pllM_out	pllM_2x_out	pllP_out	int_pllA_out	Have super clock divider ? (# of bits)	Have NV divider? (# of bits)
pllC_out1		x							8
pllM_out1				x					8
pllP_out1						x			8
pllP_out2						x			8
pllP_out3						x			8
pllP_out4						x			8
pllA_out0							x		8

6.2 Clocking Architecture

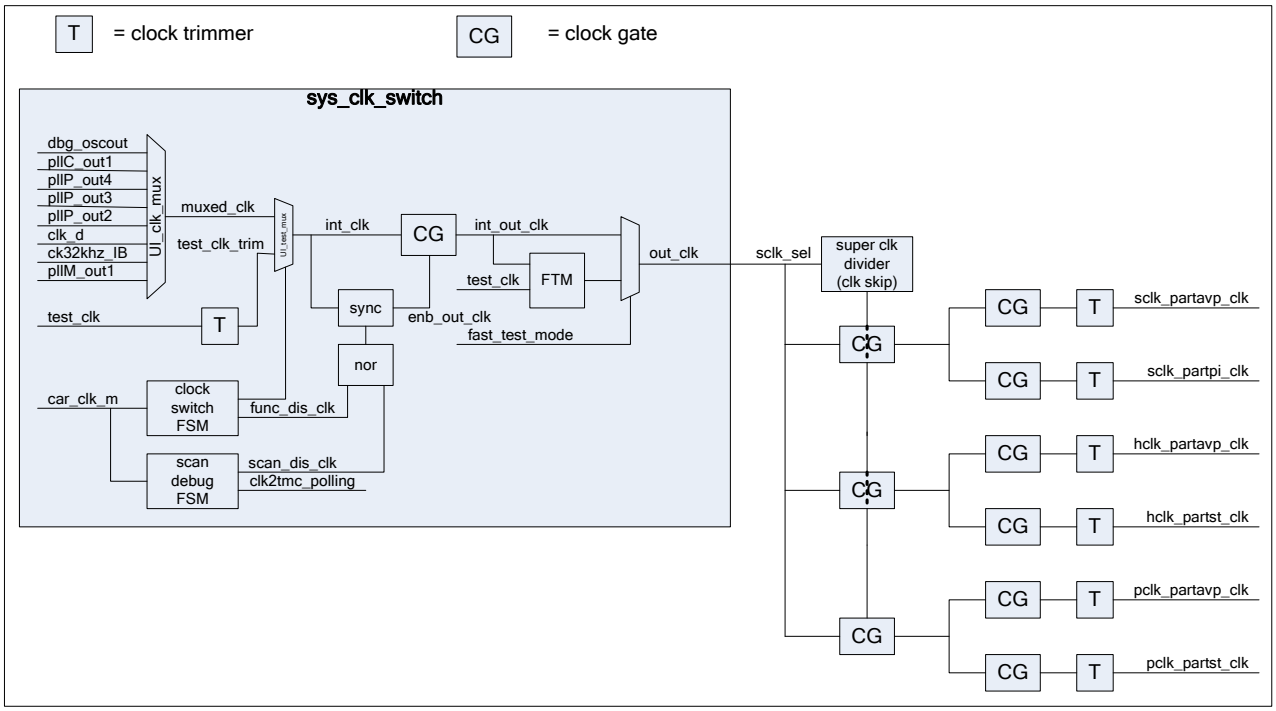
6.2.1 Main Clock Sources



6.2.2 G CPU Clock Structure (in mini-CAR CPU)



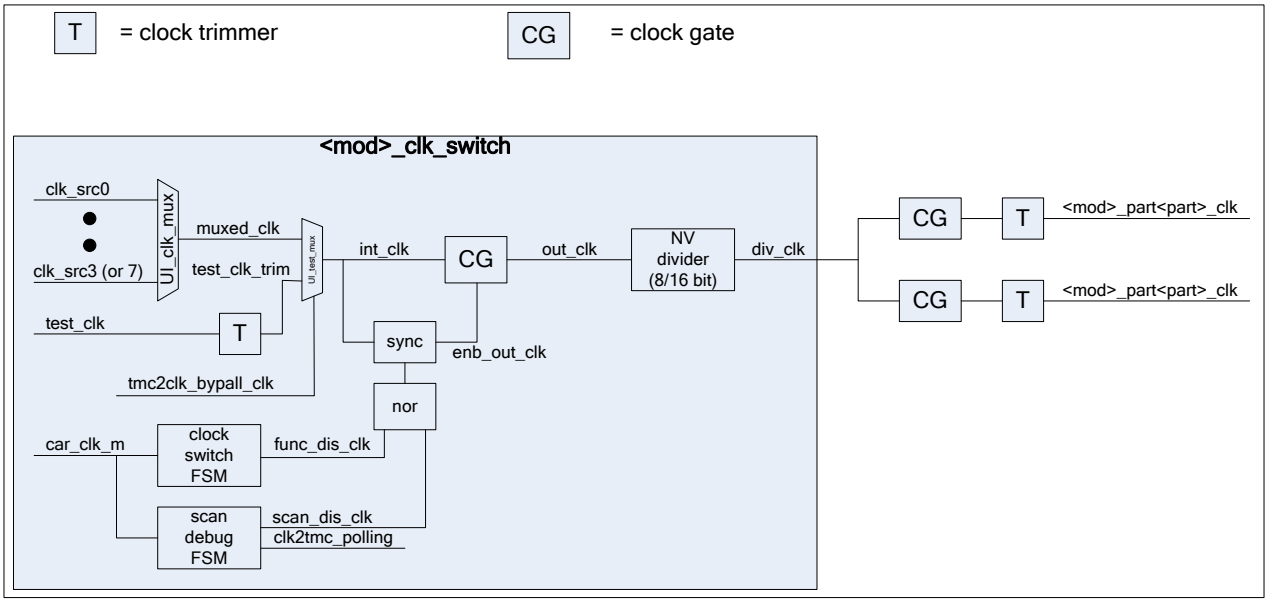
6.2.3 System Clock Structure (in CAR Center)



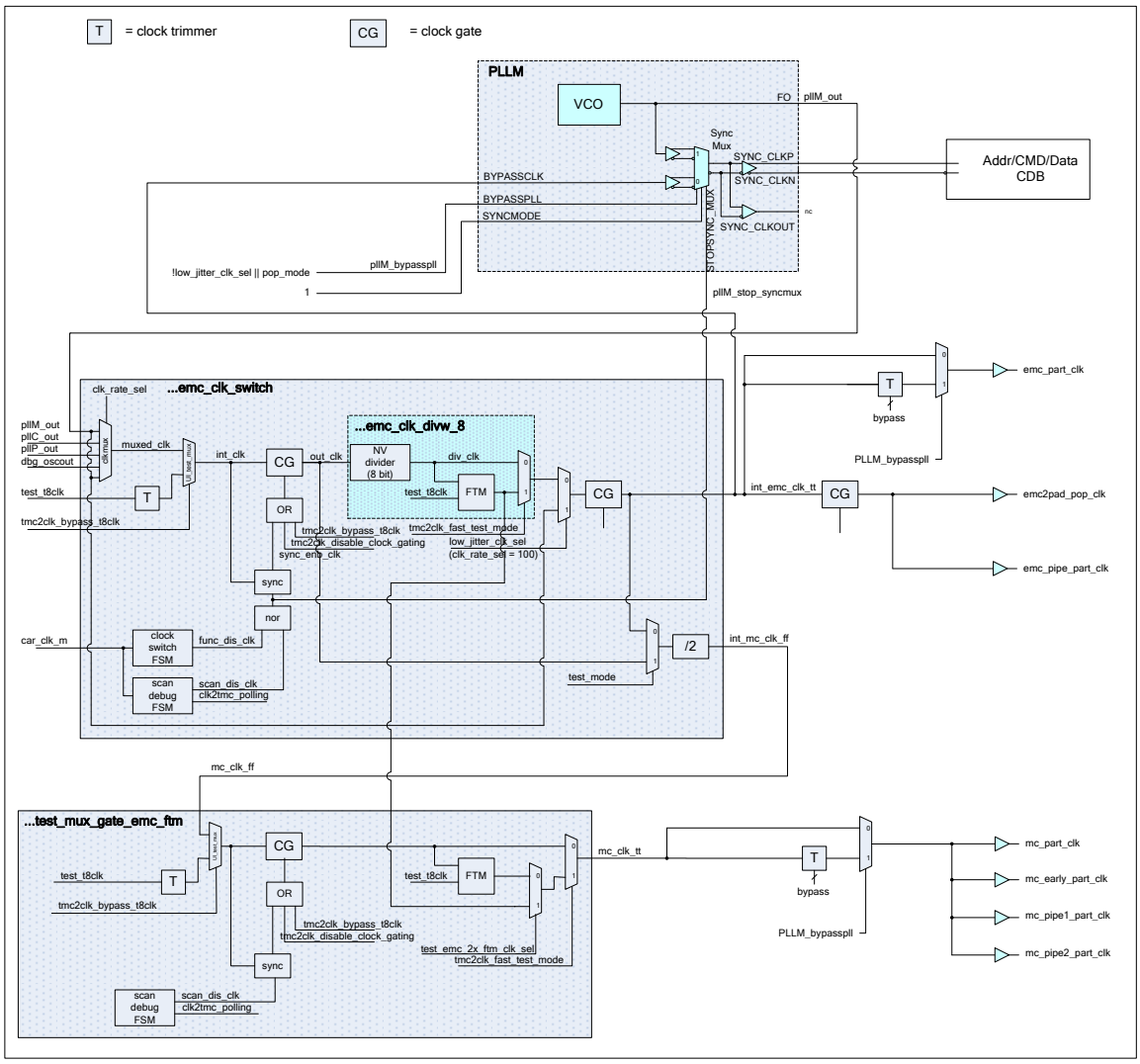
Here, “sclk” is the system clock used by the system/COP, “hclk” is the AHB bus clock, and “pclk” is the APB bus clock.

6.3 Peripheral Clock Structure (in CAR Center)

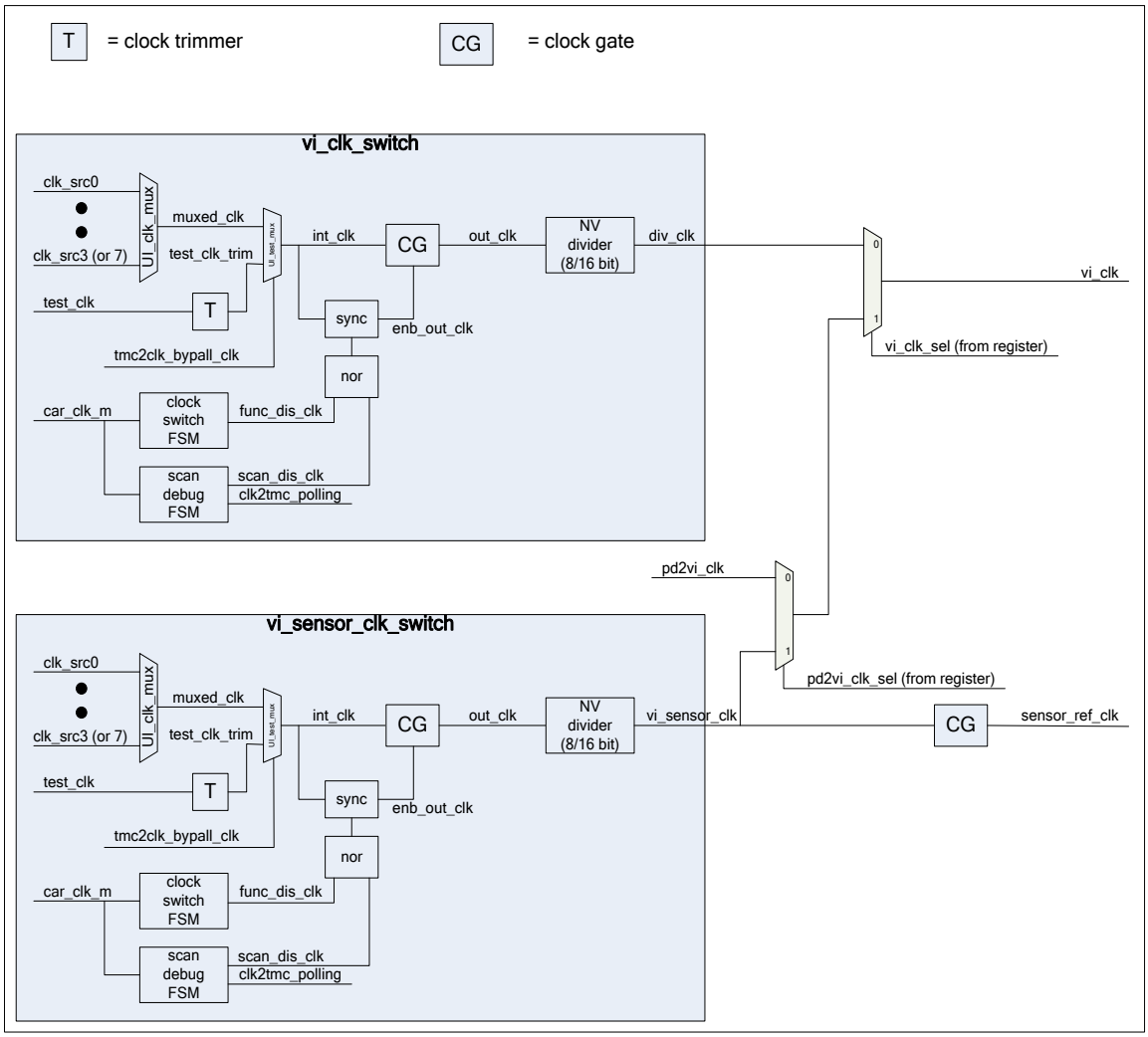
6.3.1 Generic



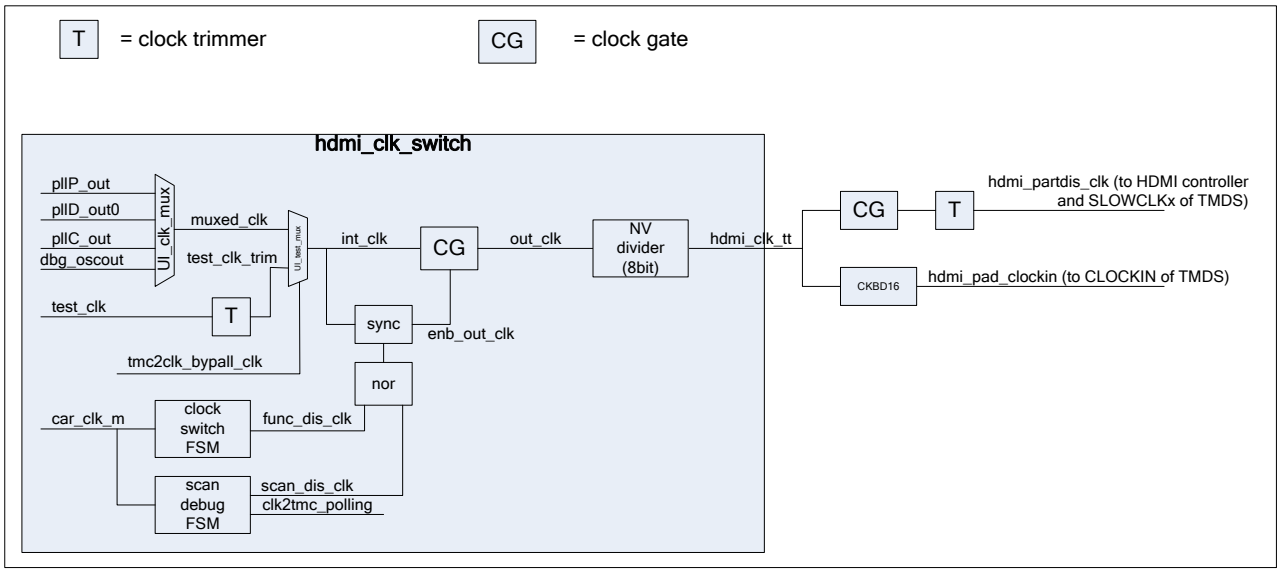
6.3.2 EMC


T = clock trimmer
CG = clock gate
PLLM
VCO
FO
pllM_out
Sync Mux
SYNC_CLKP
SYNC_CLKN
SYNC_CLKOUT
STOPSYNC_MUX
BYPASSCLK
BYPASSPLL
SYNCMODE
Addr/CMD/Data CDB
!low_jitter_clk_sel || pop_mode
pllM_bypasspll
1
pllM_stop_syncmux
clk_rate_sel
...emc_clk_switch
pllM_out
pllC_out
pllP_out
dbg_oscout
clk_mux
muxed_clk
int_clk
CG
out_clk
...emc_clk_divw_8
NV divider (8 bit)
div_clk
test_t8clk
FTM
OR
tmc2clk_bypass_t8clk
tmc2clk_disable_clock_gating
tmc2clk_fast_test_mode
low_jitter_clk_sel
(clk_rate_sel = 100)
sync_emc_clk
sync
nor
clock switch FSM
func_dis_clk
scan debug FSM
scan_dis_clk
clk2tmc_polling
car_clk_m
test_mode
/2
int_mc_clk_ff
int_emc_clk_tt
T
bypass
PLLM_bypasspll
emc_part_clk
emc2pad_pop_clk
emc_pipe_part_clk
mc_clk_ff
...test_mux_gate_emc_ftm
test_emc_2x_ftm_clk_sel
mc_clk_tt
mc_part_clk
mc_early_part_clk
mc_pipe1_part_clk
mc_pipe2_part_clk

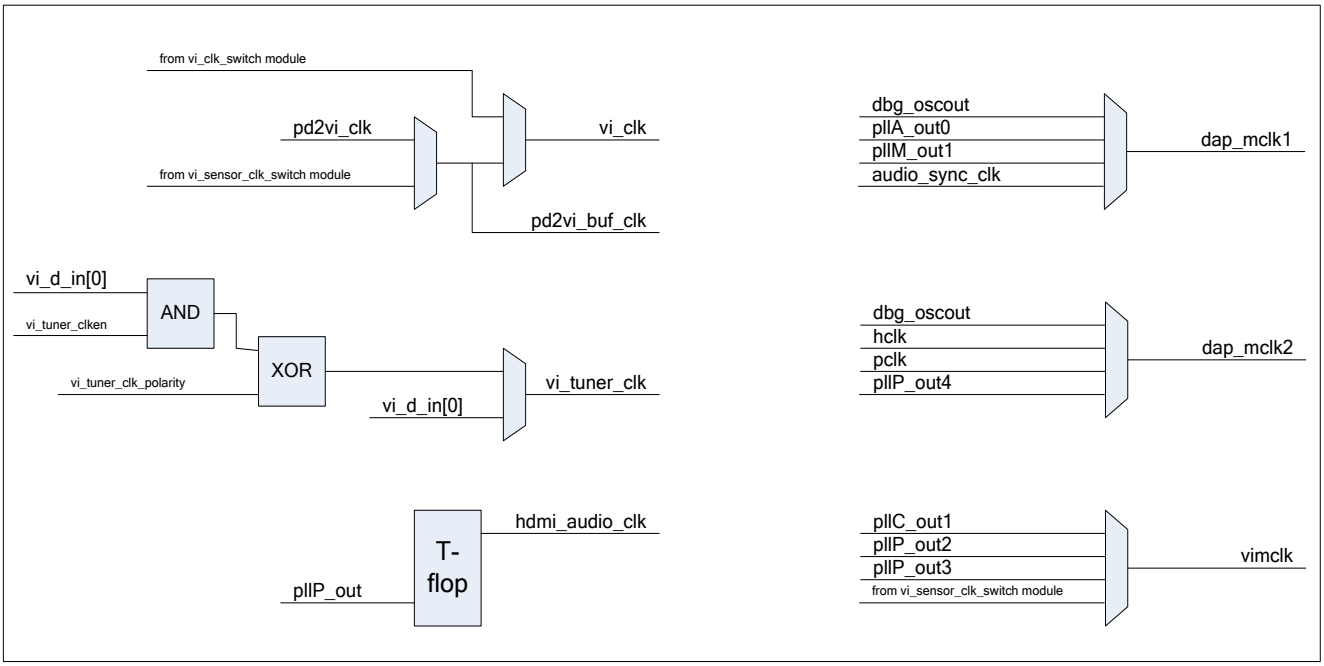
6.3.3 VI/VI Sensor



6.3.4 HDMI



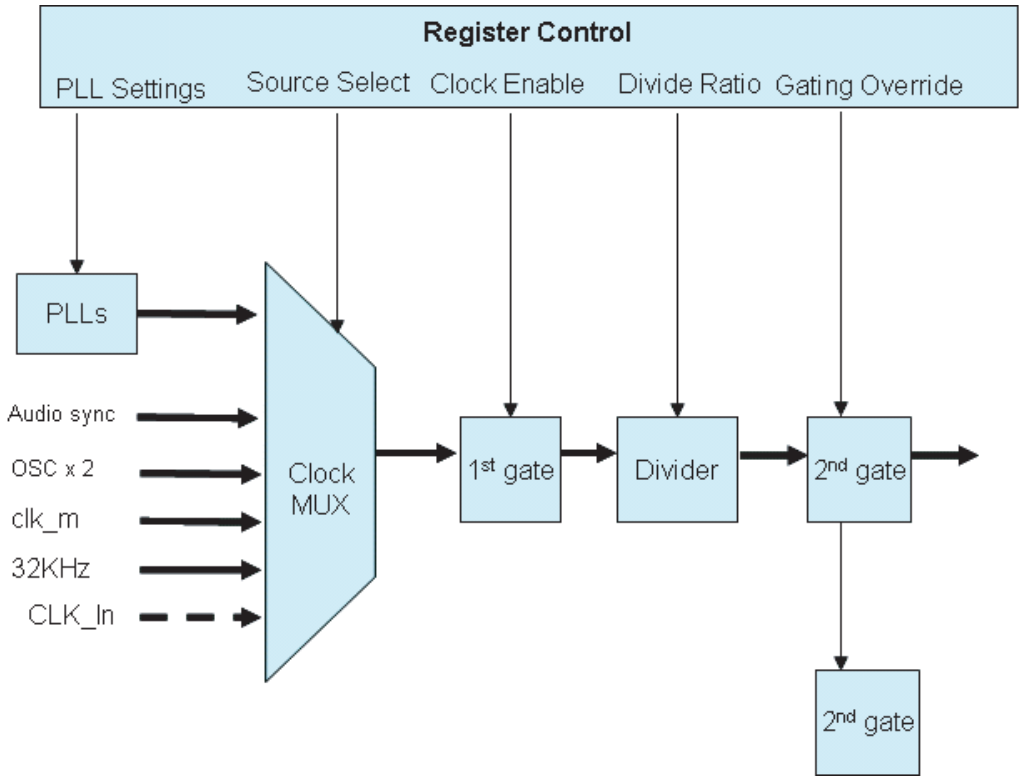
6.3.5 Miscellaneous Peripheral Clocks



6.4 Software Features and Programming Model

6.4.1 Clock Control

The figure below illustrates the software clock control model.



6.4.2 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.

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.
- Change the module clock sources for all modules that will use the new PLL settings.

6.4.3 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 = (ddddddddddddddddd + 1)

6.4.4 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" but incurs a delay of 400 ns to 600 ns. You must only write to the module's clock source register (See Clock Gating (Override)). The audio_sync_clk must be set up in advance and selected as sources before the devices that will use that source are enabled.

The clock divider to a running module can also be changed by "glitch free". 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.

Note that changing the clock source and divider simultaneously to a running module will have undefined results: hardware cannot guarantee which one will change first. To change both (either after system reset or for any other reason), follow one of these sequences:

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 to the module
4. Change the clock source to the module
5. Wait 2 us for the clock to flush through the pipe / logic.
6. Deassert reset to the module.

Sequence 2 to change clock source and divider:

1. Calculate maximum target frequency at which the device can run given the current divisor and desired clock source, and change the clock source so as not to exceed the device's maximum frequency, OR
2. Calculate the maximum target frequency at which the device can run given the current clock source and desired divisor, and change the divisor so as not to exceed the device's maximum frequency,
3. Then wait 2 us before changing the divisor (first case above) or clock source (second case above).

6.4.5 Clock Gating (Override)

Please refer to the register definition section for more information.

6.4.5.1 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.

6.4.6 Use-Case Restrictions

6.4.6.1 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 12 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 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 should be to use either PLLD or PLLC, depending on the trade-off that is acceptable. If PLLD is used, power consumption is 5x higher than PLLC. If PLLC is used, it may compromise some 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 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 13. 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

6.5 Clock and Reset Controller Registers

Since this chip is the system controller, resets are now 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 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 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™ controller

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 de-assert reset to the boot blocks first and extend the resets to the MPCore/ARM7™ processor for another 511 oscillator clock periods. This will eliminate the chance of either processors trying to talk to a boot device while it is still in 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.

Precaution

- All modules support changing the clock divider ratio without disabling the clock.
- All modules' clock switching is glitch free except "audio_sync_clk".
- For "audio_sync_clk", the clock source select needs to be setup before changing the device clock source to use that audio clock or to enable the device which use that audio clock.
- Before stopping clock (via CLK_OUT_ENB_L/H/U registers) and/or asserting reset (via RST_DEVICES_L/H/U registers) to a module, it's very important to first check the module to make sure it's 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 setup 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 registers).
2. Enable clock to the device (via CLK_OUT_ENB_L/H/U registers).
3. Change the clock divisor to the device (via CLK_SOURCE_ register).
4. Wait 1us to make sure clock divider has changed.
5. Change the clock source to the device (via CLK_SOURCE_ register).
6. Wait 2us to make sure clock source/device logic is stabilized.
7. Deassert device's reset (via RST_DEVICES_L/H/U 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 registers).
3. Make sure clock to the device is enabled (via CLK_OUT_ENB_L/H/U registers).
4. Change the clock divisor to the device (via CLK_SOURCE_ register).
5. Wait 1us to make sure clock divider has changed.
6. Change the clock source to the device (via CLK_SOURCE_ register).
7. Wait 2us to make sure clock source/device logic is stabilized.
8. Deassert device's reset (via RST_DEVICES_L/H/U registers).

(B) Method 2 -- (not using reset).

1. Make sure clock to the device is enabled (via CLK_OUT_ENB_L/H/U registers).

2. Depending on the max rated frequency of the device, and the current and target clock source/divider value, either change the divider first or the clock source first so that it won't create a temporary situation where the max frequency for that device is violated. Wait for 1us between changing the divider and clock source programming.
3. Wait 2us to make sure 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 registers).
2. Wait 2us to make sure device logic is stabilized.
3. Deassert device's reset (via RST_DEVICES_L/H/U registers).

Method used to wait for 1 or 2 usec mentioned above.

(A) To use one of the four timers, please refer to the Timer section of this document.

1. Program TMR_PTV register's TMR_PTV field with desired USEC count.
2. Program TMR_PTV register's EN field to enable timer.
3. Poll TMR_PCR register's TMR_PCV field until it reaches 0 to indicate the USEC count has been reached.
4. Program TMR_PTV register's EN field to disable timer.

Main clock sources used by the system:

(A) Primary clocks.

- "osc" or "clk_m" which can be either 12 MHz or 26 MHz.
- "clk_s" which is 32 KHz.

(B) Derived clocks.

- "clk_d" which is clock doubled from "clk_m".

(C) PLL clocks.

- "PLLC" which is general purpose and its output is called "pllC_out0".
- "PLLM" (memory) which is general purpose and its output is called "pllM_out0".
- "PLLP" (peripheral) which is fixed at 408MHz and its output is called "pllP_out0".
- "PLLA" (audio) which is cascaded from PLLP and is used for audio purposes.
- "PLLU" (USB) which has 3 outputs and fixed at 12MHz, 60MHz, and 480MHz.
- "PLLD" (si) which can go up-to 1GHz and its output (after a fix /2) is called "pllD_out0".
- "PLLX" which is extra high frequency used only by Cortex® processors and is called "pllX_out0".
- "PLLE" which is only used by PCIE

(D) PLL divided down clocks (each divider has 7-integer bit 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" and is fixed at 9.6 MHz.
- "pllP_out2" is divided-down from "pllP_out0" and is fixed at 48.0 MHz.
- "pllP_out3" is divided-down from "pllP_out0" and is fixed at 102.0 MHz.
- "pllP_out4" is divided-down from "pllP_out0" and is fixed at 102.0 MHz.
- "pllA_out0" is divided-down from "pllA" output.

Note: With the exception of PLLA, the other 7 PLLs all use "osc" as reference clock.

Multi-Address:

Register/Bit marked with Multi-Address tag denotes multiple addresses can access same register/bit.

CCLK Multi-Address (Legacy, CPUG and CPULP Multiple Address Registers and Bits):

For CCLK (CPU_CLK) related registers values:

- Three sets of addresses are provided to support legacy, and G+LP CPU clusters.
- Two sets of HW flops are used to store programmed values for G and LP CPU clusters.
- New addresses added for CPU G and LP clusters in Tegra 3.
- Original legacy address always read and write active cluster flop value.
- New address read and write corresponding G or LP flop value.
- In some cases, only CPU bit in an otherwise single access register is multi-address accessible. ex. CLK_ENB_CPU.

FUSE Multi-Address: Fuse and CAR Multiple Address Registers

- 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 reg wr

WatchDog (deadman) Timer

The purpose of the watchdog timer is to recover from hang/lockup condition by resetting either system and/or COP and/or CPU. Either timer1 or timer2 can be used as watchdog timer. Please refer to the Timer section of this document for more information on watchdog timer usage.

6.5.1 CLK_RST_CONTROLLER_RST_SOURCE_0

Offset: 000h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx00x000

Bit	R/W	Reset	Description
19	RO	X	WDT_CPU3_RST_STA: CPU3 reset by watch dog timer (RO
18	RO	X	WDT_CPU2_RST_STA: CPU2 reset by watch dog timer (RO
17	RO	X	WDT_CPU1_RST_STA: CPU1 reset by watch dog timer (RO
16	RO	X	WDT_CPU0_RST_STA: CPU0 reset by watch dog timer (RO)
13	RO	X	SWR_SYS_RST_STA: System reset by SW (RO)
12	RO	X	WDT_SYS_RST_STA: System reset by watch dog timer (RO)
11	RO	X	SWR_COP_RST_STA: COP reset by SW (RO)
10	RO	X	WDT_COP_RST_STA: COP reset by watch dog timer (RO)
9	RO	X	SWR_CPU_RST_STA: CPU reset by SW (RO)
8	RO	X	WDT_CPU_RST_STA: CPU reset by watch dog timer (RO)

Bit	R/W	Reset	Description
5	RW	DISABLE	WDT_EN: Enable Watch Dog Timer (Dead Man Timer) 0 = DISABLE 1 = ENABLE
4	RW	TIMER1	WDT_SEL: Watch Dog Timer Select 0 = TIMER1 1 = TIMER2
2	RW	DISABLE	WDT_SYS_RST_EN: Enable Watch Dog Timer reset for system. 0 = DISABLE 1 = ENABLE
1	RW	DISABLE	WDT_COP_RST_EN: Enable Watch Dog Timer reset for COP 0 = DISABLE 1 = ENABLE
0	RW	DISABLE	WDT_CPU_RST_EN: Enable Watch Dog Timer reset for CPU 0 = DISABLE 1 = ENABLE

6.5.2 CLK_RST_CONTROLLER_RST_DEVICES_L_0

Offset: 004h | Read/Write: R/W | Reset: 0b011111x11111111111111110110xx01

Bit	Reset	Description
31	DISABLE	SWR_CACHE2_RST: Reset COP cache controller. 0 = DISABLE 1 = ENABLE
30	ENABLE	SWR_I2S0_RST: Reset I2S 0 Controller 0 = DISABLE 1 = ENABLE
29	ENABLE	SWR_VCP_RST: Reset vector co-processor. 0 = DISABLE 1 = ENABLE
28	ENABLE	SWR_HOST1X_RST: Reset HOST1X. 0 = DISABLE 1 = ENABLE
27	ENABLE	SWR_DISP1_RST: Reset DISP1 controller. 0 = DISABLE 1 = ENABLE
26	ENABLE	SWR_DISP2_RST: Reset DISP2 controller. 0 = DISABLE 1 = ENABLE
24	ENABLE	SWR_3D_RST: Reset 3D controller. 0 = DISABLE 1 = ENABLE
23	ENABLE	SWR_ISP_RST: Reset ISP controller. 0 = DISABLE 1 = ENABLE
22	ENABLE	SWR_USBD_RST: Reset USB controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
21	ENABLE	SWR_2D_RST: Reset 2D graphics engine controller. 0 = DISABLE 1 = ENABLE
20	ENABLE	SWR_VI_RST: Reset VI controller. 0 = DISABLE 1 = ENABLE
19	ENABLE	SWR_EPP_RST: Reset EPP controller. 0 = DISABLE 1 = ENABLE
18	ENABLE	SWR_I2S2_RST: Reset I2S 2 Controller 0 = DISABLE 1 = ENABLE
17	ENABLE	SWR_PWM_RST: Reset Pulse Width Modulator 0 = DISABLE 1 = ENABLE
15	ENABLE	SWR_SDMMC4_RST: Reset SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	ENABLE	SWR_SDMMC1_RST: Reset SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	ENABLE	SWR_NDFLASH_RST: Reset NAND flash controller. 0 = DISABLE 1 = ENABLE
12	ENABLE	SWR_I2C1_RST: Reset I2C1 Controller 0 = DISABLE 1 = ENABLE
11	ENABLE	SWR_I2S1_RST: Reset I2S 1 Controller 0 = DISABLE 1 = ENABLE
10	ENABLE	SWR_SPDIF_RST: Reset SPDIF Controller 0 = DISABLE 1 = ENABLE
9	ENABLE	SWR_SDMMC2_RST: Reset SDMMC2 Controller 0 = DISABLE 1 = ENABLE
8	DISABLE	SWR_GPIO_RST: Reset GPIO Controller 0 = DISABLE 1 = ENABLE
7	ENABLE	SWR_UART2_RST: Reset UARTB/VFIR Controller 0 = DISABLE 1 = ENABLE
6	ENABLE	SWR_UART1_RST: Reset UARTA Controller 0 = DISABLE 1 = ENABLE
5	DISABLE	SWR_TMR_RST: Reset Timer Controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	DISABLE	SWR_TRIG_SYS_RST: Write 1 to pulse System Reset Signal. HW clears this bit 0 = DISABLE 1 = ENABLE
1	DISABLE	SWR_COP_RST: Write 1 to force COP Reset Signal. SW needs to clear this bit when done. 0 = DISABLE 1 = ENABLE
0	ENABLE	SWR_CPU_RST: Write 1 to force CPU Reset Signal. SW needs to clear this bit when done. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE

6.5.3 CLK_RST_CONTROLLER_RST_DEVICES_H_0

Offset: 008h | Read/Write: R/W | Reset: 0b1111111x111111111111x0110x1xx111

Bit	Reset	Description
31	ENABLE	SWR_BSEV_RST: Reset BSEV controller. 0 = DISABLE 1 = ENABLE
30	ENABLE	SWR_BSEA_RST: Reset BSEA controller. 0 = DISABLE 1 = ENABLE
29	ENABLE	SWR_VDE_RST: Reset VDE & BSEV controller. 0 = DISABLE 1 = ENABLE
28	ENABLE	SWR_MPE_RST: Reset MPE controller. 0 = DISABLE 1 = ENABLE
27	ENABLE	SWR_USB3_RST: Reset USB3 controller. 0 = DISABLE 1 = ENABLE
26	ENABLE	SWR_USB2_RST: Reset USB2 controller. 0 = DISABLE 1 = ENABLE
25	ENABLE	SWR_EMC_RST: Reset EMC controller. 0 = DISABLE 1 = ENABLE
23	ENABLE	SWR_UART3_RST: Reset UARTC Controller 0 = DISABLE 1 = ENABLE
22	ENABLE	SWR_I2C2_RST: Reset I2C 2 controller. 0 = DISABLE 1 = ENABLE
21	ENABLE	SWR_TVDAC_RST: Reset TVDAC controller. 0 = DISABLE 1 = ENABLE
20	ENABLE	SWR_CSI_RST: Reset CSI controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19	ENABLE	SWR_HDMI_RST: Reset HDMI 0 = DISABLE 1 = ENABLE
18	ENABLE	SWR_MIPI_RST: Reset MIPI base-band HSI controller. 0 = DISABLE 1 = ENABLE
17	ENABLE	SWR_TVO_RST: Reset TVO/CVE controller 0 = DISABLE 1 = ENABLE
16	ENABLE	SWR_DSI_RST: Reset DSI controller 0 = DISABLE 1 = ENABLE
15	ENABLE	SWR_DVC_I2C_RST: Reset DVC-I2C Controller 0 = DISABLE 1 = ENABLE
14	ENABLE	SWR_SBC3_RST: Reset SBC 3 Controller. 0 = DISABLE 1 = ENABLE
12	ENABLE	SWR_SBC2_RST: Reset SBC 2 Controller. 0 = DISABLE 1 = ENABLE
10	DISABLE	SWR_SNOR_RST: Reset NOR Flash Controller. 0 = DISABLE 1 = ENABLE
9	ENABLE	SWR_SBC1_RST: Reset SBC 1 Controller. 0 = DISABLE 1 = ENABLE
8	ENABLE	SWR_KFUSE_RST: Reset KFuse controller. 0 = DISABLE 1 = ENABLE
5	ENABLE	SWR_STAT_MON_RST: Reset statistic monitor 0 = DISABLE 1 = ENABLE
2	ENABLE	SWR_APBDMA_RST: Reset APB-DMA. 0 = DISABLE 1 = ENABLE
1	ENABLE	SWR_AHBDMA_RST: Reset AHB-DMA. 0 = DISABLE 1 = ENABLE
0	ENABLE	SWR_MEM_RST: Reset MC. 0 = DISABLE 1 = ENABLE

6.5.4 CLK_RST_CONTROLLER_RST_DEVICES_U_0

Offset: 00ch | Read/Write: R/W | Reset: 0b1111xx1010111111111

Bit	Reset	Description
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
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
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	ENABLE	SWR_UART5_RST: Reset UARTE controller. 0 = DISABLE 1 = ENABLE
1	ENABLE	SWR_UART4_RST: Reset UARTD controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	ENABLE	SWR_SPEEDO_RST: Reset SPEEDO controller. 0 = DISABLE 1 = ENABLE

6.5.5 CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0

Offset: 010h | Read/Write: R/W | Reset: 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 co-processor. 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
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 I2S 2 controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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 SPDIF 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_UART2: Enable clock to UARTB/VFIR Controller 0 = DISABLE 1 = ENABLE
6	DISABLE	CLK_ENB_UART1: Enable clock to UARTA Controller 0 = DISABLE 1 = ENABLE
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

6.5.6 CLK_RST_CONTROLLER_CLK_OUT_ENB_H_0

Offset: 014h | Read/Write: R/W | Reset: 0b0000000x000000000000x1001000x000

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	DISABLE	CLK_ENB_MPE: Enable clock to MPE controller. 0 = DISABLE 1 = ENABLE
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
23	DISABLE	CLK_ENB_UART3: Enable clock to UARTC Controller 0 = DISABLE 1 = ENABLE
22	DISABLE	CLK_ENB_I2C2: Enable clock to I2C2 controller. 0 = DISABLE 1 = ENABLE
21	DISABLE	CLK_ENB_TVDAC: Enable clock to TVDAC 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
18	DISABLE	CLK_ENB_MIPI: Enable clock to MIPI base-band HSI controller 0 = DISABLE 1 = ENABLE
17	DISABLE	CLK_ENB_TVO: Enable clock to TVO/CVE controller 0 = DISABLE 1 = ENABLE
16	DISABLE	CLK_ENB_DSI: Enable clock to DSI controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
15	DISABLE	CLK_ENB_DVC_I2C: Enable clock to DVC-I2C Controller 0 = DISABLE 1 = ENABLE
14	DISABLE	CLK_ENB_SBC3: Enable clock to SBC 3 Controller. 0 = DISABLE 1 = ENABLE
12	DISABLE	CLK_ENB_SBC2: Enable clock to SBC 2 Controller. 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 SBC 1 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

6.5.7 CLK_RST_CONTROLLER_CLK_OUT_ENB_U_0

Offset: 018h | Read/Write: R/W | Reset: 0b0x1x11111x0000xx01x1000000000

Bit	Reset	Description
28	DISABLE	CLK_ENB_SUS_OUT: Enable clock to SUS pad. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26	ENABLE	CLK_M_DOUBLER_ENB: Enable CLK_M clk doubler. 0 = DISABLE 1 = ENABLE
24	ENABLE	CLK_ENB_CRAM2: Enable COP cache RAM clk. 0 = DISABLE 1 = ENABLE
23	ENABLE	CLK_ENB_IRAMD: Enable IRAMD clk. 0 = DISABLE 1 = ENABLE
22	ENABLE	CLK_ENB_IRAMC: Enable IRAMC clk. 0 = DISABLE 1 = ENABLE
21	ENABLE	CLK_ENB_IRAMB: Enable IRAMB clk. 0 = DISABLE 1 = ENABLE
20	ENABLE	CLK_ENB_IRAMA: Enable IRAMA clk. 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
15	DISABLE	CLK_ENB_DTV: Enable clock to DTV Controller. 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

Bit	Reset	Description
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	DISABLE	CLK_ENB_UART5: Enable clock to UARTE. 0 = DISABLE 1 = ENABLE
1	DISABLE	CLK_ENB_UART4: Enable clock to UARTD. 0 = DISABLE 1 = ENABLE
0	DISABLE	CLK_ENB_SPEEDO: Enable clock to SPEEDO. 0 = DISABLE 1 = ENABLE

6.5.8 CLK_RST_CONTROLLER_CCLK_BURST_POLICY_0

CPU (CCLK) and COP/System (SCLK) Clock Control

Here, MPCore is referred to as CPU while ARM7 is referred to as COP. The clock control mechanism for CPU and COP/system are identical. Each CCLK and SCLK clock domain can have 5 states. They are SUSP (suspend) where the clock source is 32KHz, and normal states (IDLE, RUN, IRQ, FIQ). Each of the normal states can be selected by SW 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 HW will jump from any state to IRQ or to FIQ automatically. Of course, if the source is from a PLL, SW 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:

- SW can simply just keep changing the clock source to one state (i.e. just CWAKEUP_IDLE_SOURCE) without changing the CPU_STATE field.
- SW can have the concept of multiple states by first setup CWAKEUP__SOURCE and then just keep changing CPU_STATE field.
- SW can enable HW auto detect of IRQ/FIQ to jump to IRQ or FIQ state.

> **Note:** Whenever clock source is switched, there is about a 400-600 ns time where clock will be stopped.

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 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 bit 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:

- It has no clock source switching penalty. SW can just pick a PLL output as a max frequency source via CCLK/SCLK_BURST_POLICY and just keep changing SUPER_CCLK/SCLK_DIVIDER to yield the desired lower "effective" frequency.

- For application where the "osc" or "osc*2" frequency is more than sufficient to do the job, PLLs can be turned off to save power and super clock divider can further divide down the "osc" or "osc*2" 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 routine.

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 simply be the same as the input frequency. In other words, there will be no clock skip or divide down.

Offset: 020h | Read/Write: R/W | Reset: 0b00010000xxxxxxxx0000000000000000

Bit	R/W	Reset	Description
31:28	RW	0x1	CPU_STATE: 0000=32KHz 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 temp 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 = pllP_out3, 0111 = clk_d, 1xxx = PLLX_out0, 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0
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 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0

Bit	R/W	Reset	Description
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 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0
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 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0

6.5.9 CLK_RST_CONTROLLER_SUPER_CCLK_DIVIDER_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 024h | Read/Write: R/W | Reset: 0b0xxx00000000000000000000000000

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
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 denote 0.5x)
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_CDIV_DIVISOR: Actual value = n + 1.

6.5.10 CLK_RST_CONTROLLER_SCLK_BURST_POLICY_0

Offset: 028h | Read/Write: R/W | Reset: 0b00010000xxxxxxxxx000x000x000x000

Bit	Reset	Description
31:28	0x1	SYS_STATE: 0000=32KHz 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

Bit	Reset	Description
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
14:12	0x0	SWAKEUP_FIQ_SOURCE: 000 = clk_m, 001 = pllC_out1, 010 = pllP_out4, 011 = pllP_out3, 100 = pllP_out2, 101 = clk_d, 110 = clk_s, 111 = pllM_out1, 0 = CLKM 1 = PLLC_OUT1 2 = PLLP_OUT4 3 = PLLP_OUT3 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_OUT3 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_OUT3 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_OUT3 4 = PLLP_OUT2 5 = CLKD 6 = CLKS 7 = PLLM_OUT1

6.5.11 CLK_RST_CONTROLLER_SUPER_SCLK_DIVIDER_0

Offset: 02ch | Read/Write: R/W | Reset: 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.

Bit	Reset	Description
15:8	0x0	SUPER_SDIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_SDIV_DIVISOR: Actual value = n + 1.

6.5.12 CLK_RST_CONTROLLER_CLK_SYSTEM_RATE_0

Audio Sync Clock

The purpose of the audio sync clock is to synchronize communication between 2 audio devices so that it will not get into a FIFO overrun/underrun problem 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 we are transmitting 44.1 KHz sample rate to I2S, SPDIFIN receive FIFO can get overrun while I2S transmit FIFO can get underrun.

Note: There's no clock switching protection for audio sync clock so one needs to setup the desired clock source before enabling the audio transmit/receive device.

HCLK/PCLK

SCLK is the main system clock of the Tegra 3 devices which can run up-to 275MHz.

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: 030h | Read/Write: R/W | Reset: 0b0x000x00

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.

6.5.13 CLK_RST_CONTROLLER_PROG_DLY_CLK_0

Clock Doubler

There are 2 clock doublers used by clock controller and PROG_DLY_CLK is used to provide programmable delay for the clock doublers.

- Clock double from "osc" clock.
- Clock double from "audio sync clock".

Offset: 034h | Read/Write: R/W | Reset: 0b0111

Bit	Reset	Description
15:12	0x7	CLK_D_DELCLK_SEL: 16 Taps of selectable delay for CLK_M clk doubler
11:8	0x7	SYNC_CLK_DELCLK_SEL: 16 Taps of selectable delay for SYNC_CLK clk doubler

6.5.14 CLK_RST_CONTROLLER_COP_CLK_SKIP_POLICY_0

COP Clock Skip

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, HW can auto detect CPU/COP IRQ/FIQ and change the skip rate as programmed by SW.

Offset: 040h | Read/Write: R/W | Reset: 0b00000000xxxxxxxxx000x000x000x000

Bit	Reset	Description
31:28	0x0	COP_CLK_SKIP_STATE: 0000=no skip. 0001=skip base on IDLE Clock skip rate; 001X=skip base on Run clock skip rate; 01XX=skip base on IRQ Clock skip rate; 1XXX=skip base 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
14:12	0x0	COP_CLK_SKIP_RATE_FIQ: skip n/16 clock.
10:8	0x0	COP_CLK_SKIP_RATE_IRQ: Same definitions as COP_CLK_SKIP_RATE_FIQ
6:4	0x0	COP_CLK_SKIP_RATE_RUN: Same definitions as COP_CLK_SKIP_RATE_FIQ
2:0	0x0	COP_CLK_SKIP_RATE_IDLE: Same definitions as COP_CLK_SKIP_RATE_FIQ

6.5.15 CLK_RST_CONTROLLER_CLK_MASK_ARM_0

Offset: 044h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxx00xxxxxxxxxxxxxx00

Bit	R/W	Reset	Description
31	RO	X	AXI_FLUSH_DONE: 1 = CPU AXI pipe is flushed.
16	RW	0x0	CLK_MASK_CPU_HALT: WARNING: This bit must not be set in SMP mode or when fastsync FIFO feature is enabled. Deprecated. Possibly used in legacy regression tests, where fastsync FIFO is disabled
1:0	RW	0x0	CLK_MASK_COP: 00 = no clock masking. 01 = u2_nwait_r. 10 = u2_nwait_r. 11 = no clock masking.

6.5.16 CLK_RST_CONTROLLER_MISC_CLK_ENB_0

Offset: 048h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	EN_PPSB_STOPCLK: 1 = enable, 0 = disable 0 = DISABLE 1 = ENABLE

6.5.17 CLK_RST_CONTROLLER_CLK_CPU_CMPLX_0

[CCLK Multi-Address]: Refer to details in file header cpu_divN moved to mselect clock. Programmable divider not available.

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

Note: Since the CPU clock can be selected from 1-of-9 clock sources (refer to CCLK_BURST_POLICY register), it's the user's responsibility to not select a bridge divide down CPU clock ratio that will exceed 1/4 of the "MAX" CPU frequency supported.
For Tegra 3 devices, the "MAX" CPU frequency is 1.1 GHz.

Offset: 04ch | Read/Write: R/W | Reset: 0b0000xxxxxx00

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	0x3	CPU_BRIDGE_CLKDIV: Unused but available

6.5.18 CLK_RST_CONTROLLER_OSC_CTRL_0

Oscillator Control

"osc" can have either of the frequencies (12 MHz or 26 MHz) coming in, the OSC_FREQ field provides a way for SW to tell HW what frequency is the incoming clock runs at. 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. In case a frequency other than those listed is used, there is a way to override the HW auto generated PLLP parameters.

Offset: 050h | Read/Write: R/W | Reset: 0b00000000000000x00000xx111111xx01

Bit	Reset	Description
31:28	0x0	OSC_FREQ: 1000 = 12.0 MHz, 1100 = 26.00 MHz 8 = OSC12 12 = OSC26
27:26	0x0	PLL_REF_DIV: PLL reference clock divide. 00 = /1, 01 = /2, 10 = /4, 11 = reserve. 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).

6.5.19 CLK_RST_CONTROLLER_PLL_LFSR_0

Offset: 054h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	RND: Random number generated from PLL linear feedback shift register.

6.5.20 CLK_RST_CONTROLLER_OSC_FREQ_DET_0

Oscillator Frequency Detect

"osc" can have any one of the frequencies (12 MHz or 26 MHz) coming in, HW is providing a mechanism of detecting which one of the frequencies the "osc" is running at. To determine the "osc" frequency, user only needs to program the number of 32 KHz (actually, 32.768 KHz) clock periods used as a fix time window from which the "osc" will be used to increment a 16-bit counter and start the frequency detection process. Once the detection process is done, user simply just checks the 16-bit count value to determine the "osc" frequency.

OSC Frequency (MHz)	Approximate OSC_FREQ_DET_CNT (using two 32KHz period as window)
12.00	732 (0x02DC)
26.00	1587 (0x0633)

Offset: 058h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxx0000

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: Indicate the # of 32KHz clock period as window in n+1 scheme.

6.5.21 CLK_RST_CONTROLLER_OSC_FREQ_DET_STATUS_0

Offset: 05ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	OSC_FREQ_DET_BUSY: 0 = not busy, 1 = busy.
15:0	X	OSC_FREQ_DET_CNT: indicate the number of osc count within the 32KHz clock reference window.

6.5.22 CLK_RST_CONTROLLER_PLLC_BASE_0

PLL Configuration

Tegra 3 devices have 10 PLLs controllable by clock control. Please consult the PLL specification for electrical requirements.

- "PLLC/PLLP/PLLA" is of type CLKPLL700_LP_A2.
- "PLLD/PLLD2" is of type CLKPLL1G_MIPI_B.
- "PLLU" is of type CLKPLL960_USB_A3.
- "PLLMis of type CLKPLL800_LP_B1 .
- "PLLX" are of type CLKPLL12G_LP_DV_B1.

- "PLLE" is of type PLL24G_SSA_LP_CML_ESD_B1 .

In general, there are 3 requirements for each PLL that SW needs to comply with:

- 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.

Table 14 PLL Requirements for software

	CLKPLL700_LP_A2	CLKPLL800_LP_B1	CLKPLL1G_MIPI_B	CLKPLL12G_LP_DV_B1	CLKPLL960_USB_A3
Input freq. (REF)	2-31MHz	2-31MHz	2-40MHz	2-31MHz	2-40MHz
Comparison freq. (CF)	1-6MHz	1-6MHz	1-6MHz	1-6MHz	1-6MHz
VCO freq. (VCO)	20-1400MHz	20-1200MHz	40-1000MHz	20-1200MHz	480-960MHz

Table 15 Charge pump current (CPCON) for CLKPLL700_LP_A2/CLKPLL800_LP_B1/CLKPLL12G_LP_DV_B1

N-divider	Comparison freq.(CF)	Charge pump current (CPCON)
>= 200	1MHz	4uA (1000)
	2-3MHz	2uA (0100)
	4-6MHz	1uA (0010)
< 200	x	0.5uA (0001)

Table 16 Charge pump current (CPCON) for CLKPLL1G_MIPI_B/CLKPLL960_USB_A3

CPCON	Description
0000	Charge Pump 0uA
0001	Charge Pump 1uA
0010	Charge Pump 2uA, when N-divider=50
0011	Charge Pump 3uA
0100	Charge Pump 4uA, when N-divider=300
1000	Charge Pump 8uA, when N-divider=600
1100	Charge Pump 12uA, when N-divider=1000
1101	Charge Pump 13uA
1110	Charge Pump 14uA
1111	Charge Pump 15uA

CPCON setting reference from CLKPLL700_LP_A2:

N >= 200

- CompFreq 1 MHz : CPCON 4uA (1000b)
- CompFreq 2-3 MHz : CPCON 2uA (0100b)
- CompFreq 4-6 MHz : CPCON 1uA (0010b)

N < 200 : CPCON 0.5uA (0001) with special note that this may not meet lock time spec

- - 19.2 MHz setting updated
- - 38.4 and 48 MHz requires PLL_REF_DIV setting of 2 and 4. Max(DIVM)=31. PLL_REF_DIV=1 would require 32, 48 respectively

Note: 15.36 MHz setting does not meet (1MHz < CompFreq < 6MHz) stability requirement. CompFreq(15.36 MHz) = 15.36/16.
15.36 support is removed because it doesn't support all PLLD frequency requirements

Table 17. PLLP Configuration Information (reference clock is osc/clk_m and PLLP-FO is fixed at 408MHz).

Reference Frequency	12.0 MHz	26.0 MHz	Resulting Frequency
DIVN	408(0d8h)	408 (0d8h)	N/A
DIVM	12 (0ch)	26 (1ah)	N/A
DIVP	1 (0h)	1 (0h)	N/A
CPCON	8 (8h)	8 (8h)	N/A
LFCON	0 (0h)	0 (0h)	N/A
VCOCON	0 (0h)	0 (0h)	N/A
DCCON	0 (0b)	0 (0b)	N/A
OUT1 div ratio	42.5 (0dh)	42.5 (0dh)	9.6MHz.
OUT2 div ratio	8.5 (07h)	8.5 (07h)	48.0MHz.
OUT3 div ratio	4.0 (04h)	4.0 (04h)	102.0MHz.
OUT4 div ratio	4.0 (02h)	4.0 (02h)	102.0MHz.

Table 18 PLLA configuration information (reference clock is taken from PLLP_OUT1 = 28.8MHz).

Audio Frequency	56.448MHz	73.728MHz	11.2896MHz	12.2880MHz
DIVN	49 (031h)	64 (040h)	49 (031h)	64 (040h)
DIVM	25 (19h)	25 (19h)	25 (19h)	25 (19h)
DIVP	1 (0h)	1 (0h)	1 (0h)	1 (0h)
OUT0 div ratio	1.0 (0h)	1.0 (0h)	5.0 (08h)	6.0 (0ah)

Table 19 PLLU configuration information (reference clock is osc/clk_m and PLLU-FOs are fixed at 12 MHz/60 MHz/480 MHz).

Reference Frequency	12.0 MHz	26.0 MHz
DIVN	960 (3c0h)	960 (3c0h)

Reference Frequency	12.0 MHz	26.0 MHz
DIVM	12 (0ch)	26 (1ah)
CPCON	12 (ch)	12 (ch)
LFCON	1 (1h)	1 (1h)

Table 20 PLLD configuration information (reference clock is osc/clk_m and PLLD-FO is max of 1GHz).

PLLD rates of interest:

- Display/HDMI requires precise 27, 74.25, 148.5 MHz;
- For 16.8 MHz {M=14, N=495) gives FO=594MHz. Divide down to 148.5 and 74.25 MHz.
- HDA (48 MHz)
- audio/SPDIF (6.144 MHz etc.)

Reference Frequency	12.0MHz	26.0MHz
DIVN	1000(3e8h)	1000(3e8h)
DIVM	12 (0ch)	26 (1ah)
DIVP	1 (0h)	1 (0h)

Note: 1GHz is not an absolute requirement.

Table 21 PLLE configuration information (reference clock is osc/clk_m and PLLE-FO is 100 MHz).

Reference Frequency	12.0MHz (PLLE_REF_SRC==OSC_DIV)
NDIV	200 (c8h)
MDIV	1 (01h)
PLDIV	24 (18h)
PLDIV_CML	(dh)

Special Consideration when Using PLLX as a Clock Source for the CPU

PLLX is shared between G and LP CPU clusters. It is physically located at the GR power partition. The proper PLLX_FO_LP_DISABLE or PLLX_FO_G_DISABLE bit should be de-asserted to enable corresponding PLLX clock output. PLLX should not be disabled unless both CPU clusters are disabled.

Offset: 080h | Read/Write: R/W | Reset: 0b000xxxxxx000xx0000000001xxx01100

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLC_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLC_ENABLE: 0 = disable, 1 = enable. 28:28 Reserved as BASE_OVERRIDE bit in PLLP0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
29	RW	REF_ENABLE	PLLC_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLC_LOCK: 0 = not lock, 1 = lock.
22:20	RW	0x0	PLLC_DIVP: 0 = post divider (2^n).
17:8	RW	0x1	PLLC_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLC_DIVM: PLL input divider.

6.5.23 CLK_RST_CONTROLLER_PLLC_OUT_0

Offset: 084h | Read/Write: R/W | Reset: 0b00000000xxxxxx11

Bit	Reset	Description
15:8	0x0	PLLC_OUT1_RATIO: PLLC_OUT1 divider from base PLLC (lsb denote 0.5x).
1	ENABLE	PLLC_OUT1_CLKEN: PLLC_OUT1 divider clk enable. 0 = disable, 1 = 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

6.5.24 CLK_RST_CONTROLLER_PLLC_MISC_0

Offset: 08ch | Read/Write: R/W | Reset: 0b00xxxxxx00x0x0000000000100000000

Bit	Reset	Description
31	0x0	PLLC_OUT1_INV_CLK: 1 = invert PLLC_OUT1 clock.
30	0x0	PLLC_OUT1_DIV_BYP: 1 = bypass PLLC_OUT1 divider.
23:22	0x0	PLLC_PTS: Base PLLC test output select.
20	0x0	PLLC_DCCON: Base PLLC DCCON control.
18	0x0	PLLC_LOCK_ENABLE: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
17:12	0x0	PLLC_LOCK_SEL: lock select.
11:8	0x1	PLLC_CPCON: Base PLLC charge pump setup control.
7:4	0x0	PLLC_LFCON: Base PLLC loop filter setup control.
3:0	0x0	PLLC_VCOCON: Base PLLC VCO range setup control.

6.5.25 CLK_RST_CONTROLLER_PLLM_BASE_0

Offset: 090h | Read/Write: R/W | Reset: 0b000xxxxxx000xx0000000001xxx01100

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLM_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLM_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLM_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 28:28 Reserved as BASE_OVERRIDE bit in PLLP 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLM_LOCK: 0 = not lock, 1 = lock.
22:20	RW	0x0	PLLM_DIVP: 0 = post divider (2^n).
17:8	RW	0x1	PLLM_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLM_DIVM: PLL input divider.

6.5.26 CLK_RST_CONTROLLER_PLLM_OUT_0

Offset: 094h | Read/Write: R/W | Reset: 0b00000000xxxxxx11

Bit	Reset	Description
15:8	0x0	PLLM_OUT1_RATIO: PLLM_OUT1 divider from base PLLM (lsb denote 0.5x).
1	ENABLE	PLLM_OUT1_CLKEN: PLLM_OUT1 divider clk enable. 0 = disable, 1 = 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

6.5.27 CLK_RST_CONTROLLER_PLLM_MISC_0

Offset: 09ch | Read/Write: R/W | Reset: 0b00xx000000x0x0000000000100000000

Bit	Reset	Description
31	0x0	PLLM_OUT1_INV_CLK: 1 = invert PLLM_OUT1 clock.
30	0x0	PLLM_OUT1_DIV_BYP: 1 = bypass PLLM_OUT1 divider.
27:24	0x0	PLLM_SETUP: Base PLLM setup.
23:22	0x0	PLLM_PTS: Base PLLM test output select.
20	0x0	PLLM_DCCON: Base PLLM DCCON control.
18	0x0	PLLM_LOCK_ENABLE: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
17:12	0x0	PLLM_LOCK_SEL: lock select.

Bit	Reset	Description
11:8	0x1	PLLM_CPCON: Base PLLM charge pump setup control.
7:4	0x0	PLLM_LFCON: Base PLLM loop filter setup control.
3:0	0x0	PLLM_VCOCON: Base PLLM VCO range setup control.

6.5.28 CLK_RST_CONTROLLER_PLLP_BASE_0

Offset: 0a0h | Read/Write: R/W | Reset: 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. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLP_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 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.

6.5.29 CLK_RST_CONTROLLER_PLLP_OUTA_0

Offset: 0a4h | Read/Write: R/W | Reset: 0b00000000xxxxx01100000000xxxxx011

Bit	Reset	Description
31:24	0x0	PLLP_OUT2_RATIO: PLLP_OUT2 divider from base PLLP (lsb denote 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 clk enable. 0 = disable, 1 = 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 denote 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 clk enable. 0 = disable, 1 = 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

6.5.30 CLK_RST_CONTROLLER_PLLP_OUTB_0

Offset: 0a8h | Read/Write: R/W | Reset: 0b00000000xxxxx01100000000xxxxx011

Bit	Reset	Description
31:24	0x0	PLLP_OUT4_RATIO: PLLP_OUT4 divider from base PLLP (lsb denote 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 clk enable. 0 = disable, 1 = 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 denote 0.5x).
2	DISABLE	PLLP_OUT3_OVRRIDE: 0 = disallow PLLP_OUT3 ratio override, 1 = enable override. 0 = DISABLE 1 = ENABLE
1	ENABLE	PLLP_OUT3_CLKEN: PLLP_OUT3 divider clk enable. 0 = disable, 1 = 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

6.5.31 CLK_RST_CONTROLLER_PLLP_MISC_0

Offset: 0ach | Read/Write: R/W | Reset: 0b0000000000x0x0000000000100000000

Bit	Reset	Description
31	0x0	PLLP_OUT4_INV_CLK: 1 = invert PLLP_OUT4 clock.
30	0x0	PLLP_OUT3_INV_CLK: 1 = invert PLLP_OUT3 clock.
29	0x0	PLLP_OUT2_INV_CLK: 1 = invert PLLP_OUT2 clock.
28	0x0	PLLP_OUT1_INV_CLK: 1 = invert PLLP_OUT1 clock.

Bit	Reset	Description
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.
20	0x0	PLLP_DCCON: Base PLLP DCCON control.
18	0x0	PLLP_LOCK_ENABLE: 1 = enable, 0 = disable. 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.

6.5.32 CLK_RST_CONTROLLER_PLLA_BASE_0

Offset: 0b0h | Read/Write: R/W | Reset: 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. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLA_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 28:28 Reserved as BASE_OVERRIDE bit in PLLP 0 = REF_ENABLE 1 = REF_DISABLE
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.

6.5.33 CLK_RST_CONTROLLER_PLLA_OUT_0

Offset: 0b4h | Read/Write: R/W | Reset: 0b00000000xxxxxx11

Bit	Reset	Description
15:8	0x0	PLLA_OUT0_RATIO: PLLA_OUT0 divider from base PLLA (lsb denote 0.5x). 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	ENABLE	PLLA_OUT0_CLKEN: PLLA_OUT0 divider clk enable. 0 = disable, 1 = 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

6.5.34 CLK_RST_CONTROLLER_PLLA_MISC_0

Offset: 0bch | Read/Write: R/W | Reset: 0b00xxxxxx00x0x0000000000100000000

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.
20	0x0	PLLA_DCCON: Base PLLA DCCON control.
18	0x0	PLLA_LOCK_ENABLE: 1 = enable, 0 = disable. 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.

6.5.35 CLK_RST_CONTROLLER_PLLU_BASE_0

Offset: 0c0h | Read/Write: R/W | Reset: 0b000xxxx00000xx0000000001xxx01100

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLU_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLU_ENABLE: 0 = disable, 1 = enable. This bit is used only when PLLU_OVERRIDE bit is set. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLU_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 28:28 Reserved as BASE_OVERRIDE bit in PLLP 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLU_LOCK: 0 = not lock, 1 = lock.
23	RW	0x0	PLLU_CLKENABLE_ICUSB: FO_ICUSB output enable. This bit is used only when PLLU_OVERRIDE bit is set.

Bit	R/W	Reset	Description
22	RW	0x0	PLLU_CLKENABLE_HSIC: FO_HSIC output enable. This bit is used only when 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 PLLU_OVERRIDE bit is set. Otherwise, USB controllers will control this automatically. 0 = disable, 1 = enable.
20	RW	0x0	PLLU_VCO_FREQ: 0 = post-div of 2, 1 = post-div of 1.
17:8	RW	0x1	PLLU_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLU_DIVM: PLL input divider.

6.5.36 CLK_RST_CONTROLLER_PLLU_MISC_0

Offset: 0cch | Read/Write: R/W | Reset: 0b000xxxx0xxxx00000000100000000

Bit	Reset	Description
29:27	0x0	PLLU_PTS: Base PLLU test output select.
22	0x0	PLLU_LOCK_ENABLE: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
17:12	0x0	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.

6.5.37 CLK_RST_CONTROLLER_PLLD_BASE_0

Offset: 0d0h | Read/Write: R/W | Reset: 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. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLD_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 28:28 Reserved as BASE_OVERRIDE bit in PLLP 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	DSIB_CLK_SRC: 0 = PLL_D, 1 = PLL_D2 0 = PLL_D 1 = PLL_D2

Bit	R/W	Reset	Description
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.

6.5.38 CLK_RST_CONTROLLER_PLLD_MISC_0

Offset: 0dch | Read/Write: R/W | Reset: 0b00000000000000000000000100000000

Bit	Reset	Description
31	0x0	PLLD_FO_MODE: 1 = 5-125MHz, 0 = 40-1000MHz.
30	0x0	PLLD_CLKENABLE: 0 = disable, 1 = normal operation.
29:27	0x0	PLLD_PTS: Base PLLD test output select.
26:24	0x0	PLLD_LOADADJ: load pulse position adjust.
23	0x0	PLLD_DIV_RST: 0 = normal operation, 1 = reset.
22	0x0	PLLD_LOCK_ENABLE: 1 = enable, 0 = disable. 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.

6.5.39 CLK_RST_CONTROLLER_PLLX_BASE_0

Offset: 0e0h | Read/Write: R/W | Reset: 0b000xxxxxx000xx0000000001xxx01100

Bit	R/W	Reset	Description
31	RW	DISABLE	PLLX_BYPASS: 0 = no bypass, 1 = bypass. 0 = DISABLE 1 = ENABLE
30	RW	DISABLE	PLLX_ENABLE: 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	RW	REF_ENABLE	PLLX_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 28:28 Reserved as BASE_OVERRIDE bit in PLLP 0 = REF_ENABLE 1 = REF_DISABLE
27	RO	X	PLLX_LOCK: 0 = not lock, 1 = lock.
22:20	RW	0x0	PLLX_DIVP: 0 = post divider (2^n).
17:8	RW	0x1	PLLX_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLX_DIVM: PLL input divider.

6.5.40 CLK_RST_CONTROLLER_PLLX_MISC_0

Note: PLLX source to cpulp (PLLX_OUT0) is div2 by default. DIV2 is controlled by PLLX_DIV2_BYPASS_LP bit. See warning about div2 programming in CCLKLP_BURST_POLICY reg.

Offset: 0e4h | Read/Write: R/W | Reset: 0b00000000x0x0000000000100000000

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=OF
27:24	0x0	PLLX_SETUP: Base PLLX setup.
23:22	0x0	PLLX_PTS: Base PLLX test output select.
20	0x1	PLLX_DCCON: Base PLLX DCCON control.
18	0x0	PLLX_LOCK_ENABLE: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
17:12	0x0	PLLX_LOCK_SEL: lock select.
11:8	0x1	PLLX_CPCON: Base PLLX charge pump setup control.
7:4	0x0	PLLX_LFCON: Base PLLX loop filter setup control.
3:0	0x0	PLLX_VCOCON: Base PLLX VCO range setup control.

6.5.41 CLK_RST_CONTROLLER_PLLE_BASE_0

Offset: 0e8h | Read/Write: R/W | Reset: 0b00001101000110001100100000000001

Bit	Reset	Description
31	DISABLE	PLLE_ENABLE_CML: Enable CML pdivider. 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
30	DISABLE	PLLE_ENABLE: PLL enable. 0 = disable, 1 = enable. 0 = DISABLE 1 = ENABLE
29	0x0	PLLE_LOCK_OVERRIDE: Forces PLL_LOCK to 1.
28	0x0	PLLE_FDIV4B: 0 gives vclock/4, 1 gives vclock/2 clock to the interpolator logic. Normally set to zero.
27:24	0xd	PLLE_PLDIV_CML: divider control for CLOCKOUT_CML/CLOCKOUTB_CML.
23:22	0x0	PLLE_EXT_SETUP_19_18: Base PLLE setup [19:18].
21:16	0x18	PLLE_PLDIV: post divider for CLOCKOUT and SYNC_CLOCKOUT.
15:8	0xc8	PLLE_NDIV: feedback divider.
7:0	0x1	PLLE_MDIV: input divider.

6.5.42 CLK_RST_CONTROLLER_PLLE_MISC_0

Offset: 0ech | Read/Write: R/W | Reset: 0b0000000000000000xxxxx00000000000

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 is PLL_READY status: 1 = PLL finish training, 0 = PLL not finish training.
14:12	RO	X	PLLE_MON_TESTOUT: Process monitor debug output.
11	RO	X	PLLE_LOCK: 0 = not lock, 1 = lock.
10	RW	REF_ENABLE	PLLE_REF_DIS: 0 = enable reference clk, 1 = disable reference clk. 0 = REF_ENABLE 1 = REF_DISABLE
9	RW	0x0	PLLE_LOCK_ENABLE: 1 = enable, 0 = disable. 0 = DISABLE 1 = ENABLE
8	RW	0x0	PLLE_PTS: Bypass PLL (similar to PTO control of other PLL). 0 = PTO always 0 if PLLE_ENABLE=0 (SYN_CLOCKOUT=0), 0 = PTO = PLLE CLOCKIN if PLLE_ENABLE=1, 1 = PTO = PLLE FO (SYN_CLOCKOUT=VCOCLOCK/PLDIV).
7:6	RW	0x0	PLLE_KCP: Base PLLE charge pump gain control.
5:4	RW	0x0	PLLE_LFRES: Base PLLE loop filter resistor control.
3:2	RW	0x0	PLLE_EXT_SETUP_17_16: Base PLLE setup[17:16].
1	RW	0x0	PLLE_SYNC_MODE: Base PLLE sync mode (leave it at 0).
0	RW	0x0	PLLE_KVCO: Base PLLE VCO gain.

6.5.43 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 to select one of the four clock sources. The divider is typically 8-bits which consist of 7 integer bits and 1 fractional bit. 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.
In order to switch from one clock source to the next, both clock sources must be active/running.

Special PLL clock usage by certain peripheral

As mentioned previously, each peripheral has a clock source select in their corresponding CLK_SOURCE_ register. But in addition, there are a number of modules which also take in an additional fix PLL clock source for portion of their logic.

Listed below is the module/logic and the fix PLL clock source they use.

Module/Logic	Fix PLL Clock Source Used
LFSR	pllM_out0
DSI	pllP_out3
DSIB	pllP_out3
CSI	pllP_out3
I2C1	pllP_out3
I2C2	pllP_out3
I2C3	pllP_out3
UARTA	pllP_out3
UARTB	pllP_out3
UARTC	pllP_out3
UARTD	pllP_out3
UARTE	pllP_out3
HDMI (audio)	pllP_out0
PLLE(ref. clk if PLLE-REF_SRC=1)	pllP_out0

Offset: 100h | Read/Write: R/W | | Reset: 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: Obsolete. Use CLK_ENB_I2S1 to control clock enable
7:0	0x0	I2S1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.44 CLK_RST_CONTROLLER_CLK_SOURCE_I2S2_0

Offset: 104h | Read/Write: R/W | Reset: 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: Obsolete. Use CLK_ENB_I2S2 to control clock enable
7:0	0x0	I2S2_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.45 CLK_RST_CONTROLLER_CLK_SOURCE_SPDIF_OUT_0

Offset: 108h | Read/Write: R/W | Reset: 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	SPDIFOUT_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.46 CLK_RST_CONTROLLER_CLK_SOURCE_SPDIF_IN_0

Offset: 10ch | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	PLLP_OUT0	SPDIFIN_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = 1'b0 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0
7:0	0x0	SPDIFIN_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.47 CLK_RST_CONTROLLER_CLK_SOURCE_PWM_0

Offset: 110h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
29:28	CLK_M	PWM_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = clk_s 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = CLK_S 3 = CLK_M
7:0	0x0	PWM_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.48 CLK_RST_CONTROLLER_CLK_SOURCE_SBC2_0

Offset: 118h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SBC2_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	SBC2_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.49 CLK_RST_CONTROLLER_CLK_SOURCE_SBC3_0

Offset: 11ch | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SBC3_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	SBC3_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.50 CLK_RST_CONTROLLER_CLK_SOURCE_I2C1_0

Offset: 124h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
31:30	CLK_M	I2C1_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
15:0	0x0	I2C1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

6.5.51 CLK_RST_CONTROLLER_CLK_SOURCE_DVC_I2C_0

Offset: 128h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
31:30	CLK_M	DVC_I2C_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
15:0	0x0	DVC_I2C_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

6.5.52 CLK_RST_CONTROLLER_CLK_SOURCE_SBC1_0

Offset: 134h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SBC1_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	SBC1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.53 CLK_RST_CONTROLLER_CLK_SOURCE_DISP1_0

Offset: 138h | Read/Write: R/W | Reset: 0b110

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

6.5.54 CLK_RST_CONTROLLER_CLK_SOURCE_DISP2_0

Offset: 13ch | Read/Write: R/W | Reset: 0b110

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 MSB 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

6.5.55 CLK_RST_CONTROLLER_CLK_SOURCE_CVE_0

Offset: 140h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	CVE_CLK_SRC: 00 = pllP_out0 01 = pllD_out0 10 = pllC_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLD_OUT0 2 = PLLC_OUT0 3 = CLK_M
7:0	0x0	CVE_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.56 CLK_RST_CONTROLLER_CLK_SOURCE_VI_0

Offset: 148h | Read/Write: R/W | Reset: 0b00xxxx00xxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	PLLM_OUT0	VI_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = pllA_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0
25	0x0	PD2VI_CLK_SEL: 0 = pd2vi_clk, 1 = vi_sensor_clk.

Bit	Reset	Description
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 denote 0.5x)

6.5.57 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC1_0

Offset: 150h | Read/Write: R/W | Reset: 0b11xxxxxx0xxx0000xxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SDMMC1_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	SDMMC1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.58 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC2_0

Offset: 154h | Read/Write: R/W | Reset: 0b11xxxxxx0xxx0000xxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SDMMC2_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	SDMMC2_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.59 CLK_RST_CONTROLLER_CLK_SOURCE_G3D_0

Offset: 158h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
31:30	PLLM_OUT0	G3D_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = pllA_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0
15:8	0x0	G3D_IDLE_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x) if all 0's, 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 denote 0.5x)

6.5.60 CLK_RST_CONTROLLER_CLK_SOURCE_G2D_0

Offset: 15ch | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
31:30	PLLM_OUT0	G2D_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = pllA_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0
15:8	0x0	G2D_IDLE_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x) if all 0's, 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 denote 0.5x)

6.5.61 CLK_RST_CONTROLLER_CLK_SOURCE_NDFLASH_0

NAND module now uses a combination of 3 clocks: nandfst_clk, nand_clk, and nand_speed_clk

- nandfst_clk and nand_clk are controlled by CLK_SOURCE_NDFLASH reg
- nand_speed_clk is controlled by CLK_SOURCE_NAND_SPEED reg

Output of clock switch is nandfst_clk at full rate.

An additional nand_clk is programmable to full or div2 rate of nandfst_clk.

Offset: 160h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	NDFLASH_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
8	0x0	NDFLASH_DIV2_SEL: 0 = ASYNC interface mode. nand_clk = nandfst_clk 1 = DDR interface mode. nand_clk = nandfst_clk/
7:0	0x0	NDFLASH_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.62 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC4_0

Offset: 164h | Read/Write: R/W | Reset: 0b11xxxxxx0xxx0000xxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SDMMC4_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	SDMMC4_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.63 CLK_RST_CONTROLLER_CLK_SOURCE_VFIR_0

Offset: 168h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	VFIR_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	VFIR_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.64 CLK_RST_CONTROLLER_CLK_SOURCE_EPP_0

Offset: 16ch | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	PLLM_OUT0	EPP_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = pllA_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0
7:0	0x0	EPP_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.65 CLK_RST_CONTROLLER_CLK_SOURCE_MPE_0

Offset: 170h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	PLLM_out0	MPE_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = pllA_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0
7:0	0x0	MPE_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.66 CLK_RST_CONTROLLER_CLK_SOURCE_HSI_0

Offset: 174h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	HSI_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	HSI_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.67 CLK_RST_CONTROLLER_CLK_SOURCE_UARTA_0

Offset: 178h | Read/Write: R/W | Reset: 0b11xxxxx0xxxxxxxx0000000000000010

Bit	Reset	Description
31:30	CLK_M	UARTA_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m UART baud clock divisors by default come directly from UART DLM/DLL registers in 16.0 format. enable the UART_DIV_ENB to use the 15.1 divisor below 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
24	DISABLE	UARTA_DIV_ENB: 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 denote 0.5x)

6.5.68 CLK_RST_CONTROLLER_CLK_SOURCE_UARTB_0

Offset: 17ch | Read/Write: R/W | Reset: 0b11xxxxx0xxxxxxxx0000000000000010

Bit	Reset	Description
31:30	CLK_M	UARTB_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m UART baud clock divisors by default come directly from UART DLM/DLL registers in 16.0 format. enable the UART_DIV_ENB to use the 15.1 divisor below 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
24	DISABLE	UARTB_DIV_ENB: 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 denote 0.5x)

6.5.69 CLK_RST_CONTROLLER_CLK_SOURCE_HOST1X_0

Offset: 180h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
31:30	PLLM_OUT0	HOST1X_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = pllA_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0
15:8	0x0	HOST1X_IDLE_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x) if all 0's, 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 denote 0.5x)

6.5.70 CLK_RST_CONTROLLER_CLK_SOURCE_TVO_0

Offset: 188h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	TVO_CLK_SRC: 00 = pllP_out0 01 = pllD_out0 10 = pllC_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLD_OUT0 2 = PLLC_OUT0 3 = CLK_M
7:0	0x0	TVO_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.71 CLK_RST_CONTROLLER_CLK_SOURCE_HDMI_0

Offset: 18ch | Read/Write: R/W | Reset: 0b110xxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:29	CLK_M	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 used to preserve 2 MSB 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	HDMI_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.72 CLK_RST_CONTROLLER_CLK_SOURCE_TVDAC_0

Offset: 194h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	TVDAC_CLK_SRC: 00 = pllP_out0 01 = pllD_out0 10 = pllC_out0 11 = clk_m 0 = PLLP_OUT0 1 = PLLD_OUT0 2 = PLLC_OUT0 3 = CLK_M
7:0	0x0	TVDAC_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.73 CLK_RST_CONTROLLER_CLK_SOURCE_I2C2_0

Offset: 198h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
31:30	CLK_M	I2C2_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
15:0	0x0	I2C2_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

6.5.74 CLK_RST_CONTROLLER_CLK_SOURCE_EMC_0

Offset: 19ch | Read/Write: R/W | Reset: 0b110xxxxxxxxxxx0xxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	EMC_2X_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = clk_m 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M
29	0x0	USE_PLLM_UD: 1 = use un-divided PllM_out0 as clock source.
16	DISABLE	MC_EMC_SAME_FREQ: 0 = MC is 1/2 the frequency of EMC. 1 = MC is the same frequency of EMC. 0 = DISABLE 1 = ENABLE
7:0	0x0	EMC_2X_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.75 CLK_RST_CONTROLLER_CLK_SOURCE_UARTC_0

Offset: 1a0h | Read/Write: R/W | Reset: 0b11xxxxx0xxxxxxxx0000000000000010

Bit	Reset	Description
31:30	CLK_M	UARTC_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = CLK_M UART baud clock divisors by default come directly from UART DLM/DLL registers in 16.0 format. enable the UART_DIV_ENB to use the 15.1 divisor below 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
24	DISABLE	UARTC_DIV_ENB: 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 denote 0.5x)

6.5.76 CLK_RST_CONTROLLER_CLK_SOURCE_VI_SENSOR_0

Offset: 1a8h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	PLLM_OUT0	VI_SENSOR_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = pllA_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0
7:0	0x0	VI_SENSOR_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.77 CLK_RST_CONTROLLER_CLK_SOURCE_SBC4_0

Offset: 1b4h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SBC4_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	SBC4_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.78 CLK_RST_CONTROLLER_CLK_SOURCE_I2C3_0

Offset: 1b8h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
31:30	CLK_M	I2C3_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
15:0	0x0	I2C3_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

6.5.79 CLK_RST_CONTROLLER_CLK_SOURCE_SDMMC3_0

Offset: 1bch | Read/Write: R/W | Reset: 0b11xxxxxx0xxx0000xxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SDMMC3_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	SDMMC3_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.80 CLK_RST_CONTROLLER_CLK_SOURCE_UARTD_0

Offset: 1c0h | Read/Write: R/W | Reset: 0b11xxxxx0xxxxxxxx0000000000000010

Bit	Reset	Description
31:30	CLK_M	UARTD_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m UART baud clock divisors by default come directly from UART DLM/DLL registers in 16.0 format. enable the UART_DIV_ENB to use the 15.1 divisor below 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
24	DISABLE	UARTD_DIV_ENB: 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 denote 0.5x)

6.5.81 CLK_RST_CONTROLLER_CLK_SOURCE_UARTE_0

Offset: 1c4h | Read/Write: R/W | Reset: 0b11xxxxx0xxxxxxxx0000000000000010

Bit	Reset	Description
31:30	CLK_M	UARTE_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m UART baud clock divisors by default come directly from UART DLM/DLL registers in 16.0 format. enable the UART_DIV_ENB to use the 15.1 divisor below 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
24	DISABLE	UARTE_DIV_ENB: 0=use UART DLM/DLL, 1=use divisor below 0 = DISABLE 1 = ENABLE
15:0	0x2	UARTE_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.82 CLK_RST_CONTROLLER_CLK_SOURCE_VDE_0

Offset: 1c8h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	VDE_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	VDE_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.83 CLK_RST_CONTROLLER_CLK_SOURCE_OWR_0

Offset: 1cch | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	OWR_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	OWR_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.84 CLK_RST_CONTROLLER_CLK_SOURCE_NOR_0

Offset: 1d0h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SNOR_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	SNOR_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.85 CLK_RST_CONTROLLER_CLK_SOURCE_CSITE_0

Offset: 1d4h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	CSITE_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	CSITE_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.86 CLK_RST_CONTROLLER_CLK_SOURCE_I2S0_0

Offset: 1d8h | Read/Write: R/W | Reset: 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: Obsolete. Use CLK_ENB_I2S0 to control clock enable
7:0	0x0	I2S0_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.87 CLK_RST_CONTROLLER_CLK_SOURCE_DTV_0

Offset: 1dch | Read/Write: R/W | Reset: 0b0

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

6.5.88 CLK_RST_CONTROLLER_CLK_SOURCE_OSC_0

Offset: 1fch | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
28	EXT_OSC	OSC_CLK_SRC: 0 = external oscillator 0 = EXT_OSC

6.5.89 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 write. As for read, again, user can use either method to retrieve the reset/clock-enable state of each peripheral.

Offset: 300h | Read/Write: R/W | Reset: 0b011111x11111111111111110110xx001

Bit	Reset	Description
31	0x0	SET_CACHE2_RST: set reset COP cache controller. 0 = DISABLE 1 = ENABLE
30	0x1	SET_I2S0_RST: set reset I2S 0 Controller 0 = DISABLE 1 = ENABLE
29	0x1	SET_VCP_RST: set reset vector co-processor. 0 = DISABLE 1 = ENABLE
28	0x1	SET_HOST1X_RST: set reset HOST1X. 0 = DISABLE 1 = ENABLE
27	0x1	SET_DISP1_RST: set reset DISP1 controller. 0 = DISABLE 1 = ENABLE
26	0x1	SET_DISP2_RST: set reset DISP2 controller. 0 = DISABLE 1 = ENABLE
24	0x1	SET_3D_RST: set reset 3D controller. 0 = DISABLE 1 = ENABLE
23	0x1	SET_ISP_RST: set reset ISP controller. 0 = DISABLE 1 = ENABLE
22	0x1	SET_USBD_RST: set reset USB controller 0 = DISABLE 1 = ENABLE
21	0x1	SET_2D_RST: set reset 2D graphics engine controller. 0 = DISABLE 1 = ENABLE
20	0x1	SET_VI_RST: set reset VI controller. 0 = DISABLE 1 = ENABLE
19	0x1	SET_EPP_RST: set reset EPP controller. 0 = DISABLE 1 = ENABLE
18	0x1	SET_I2S2_RST: set reset I2S 2 Controller 0 = DISABLE 1 = ENABLE
17	0x1	SET_PWM_RST: set reset Pulse Width Modulator 0 = DISABLE 1 = ENABLE
15	0x1	SET_SDMMC4_RST: set reset SDMMC4 controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
14	0x1	SET_SDMMC1_RST: set reset SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	0x1	SET_NDFLASH_RST: set reset NAND flash controller. 0 = DISABLE 1 = ENABLE
12	0x1	SET_I2C1_RST: set reset I2C1 Controller 0 = DISABLE 1 = ENABLE
11	0x1	SET_I2S1_RST: set reset I2S 1 Controller 0 = DISABLE 1 = ENABLE
10	0x1	SET_SPDIF_RST: set reset SPDIF Controller 0 = DISABLE 1 = ENABLE
9	0x1	SET_SDMMC2_RST: set reset SDMMC2 Controller 0 = DISABLE 1 = ENABLE
8	0x0	SET_GPIO_RST: set reset GPIO Controller 0 = DISABLE 1 = ENABLE
7	0x1	SET_UARTB_RST: set reset UARTB/VFIR Controller 0 = DISABLE 1 = ENABLE
6	0x1	SET_UARTA_RST: set reset UARTA Controller 0 = DISABLE 1 = ENABLE
5	0x0	SET_TMR_RST: set reset Timer Controller 0 = DISABLE 1 = ENABLE
2	0x0	SET_TRIG_SYS_RST: Write 1 to pulse System Reset Signal. 0 = DISABLE 1 = ENABLE
1	0x0	SET_COP_RST: set reset COP. 0 = DISABLE 1 = ENABLE
0	0x1	SET_CPU_RST: set reset CPU. . [CCLK Multi-Address 0 = DISABLE 1 = ENABLE

6.5.90 CLK_RST_CONTROLLER_RST_DEV_L_CLR_0

Offset: 304h | Read/Write: R/W | Reset: 0b011111x11111111111111110110xxx01

Bit	Reset	Description
31	0x0	CLR_CACHE2_RST: clear reset COP cache controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
30	0x1	CLR_I2S0_RST: clear reset I2S 0 Controller 0 = DISABLE 1 = ENABLE
28	0x1	CLR_HOST1X_RST: clear reset HOST1X. 0 = DISABLE 1 = ENABLE
27	0x1	CLR_DISP1_RST: clear reset DISP1 controller. 0 = DISABLE 1 = ENABLE
26	0x1	CLR_DISP2_RST: clear reset DISP2 controller 0 = DISABLE 1 = ENABLE.
24	0x1	CLR_3D_RST: clear reset 3D controller. 0 = DISABLE 1 = ENABLE
23	0x1	CLR_ISP_RST: clear reset ISP controller. 0 = DISABLE 1 = ENABLE
22	0x1	CLR_USBD_RST: clear reset USB controller 0 = DISABLE 1 = ENABLE
21	0x1	CLR_2D_RST: clear reset 2D graphics engine controller. 0 = DISABLE 1 = ENABLE
20	0x1	CLR_VI_RST: clear reset VI controller. 0 = DISABLE 1 = ENABLE
19	0x1	CLR_EPP_RST: clear reset EPP controller. 0 = DISABLE 1 = ENABLE
18	0x1	CLR_I2S2_RST: clear reset I2S 2 Controller 0 = DISABLE 1 = ENABLE
17	0x1	CLR_PWM_RST: clear reset Pulse Width Modulator 0 = DISABLE 1 = ENABLE
15	0x1	CLR_SDMMC4_RST: clear reset SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	0x1	CLR_SDMMC1_RST: clear reset SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	0x1	CLR_NDFLASH_RST: clear reset NAND flash controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	0x1	CLR_I2C1_RST: clear reset I2C1 Controller 0 = DISABLE 1 = ENABLE
11	0x1	CLR_I2S1_RST: clear reset I2S 1 Controller 0 = DISABLE 1 = ENABLE
10	0x1	CLR_SPDIF_RST: clear reset SPDIF Controller 0 = DISABLE 1 = ENABLE
9	0x1	CLR_SDMMC2_RST: clear reset SDMMC2 Controller 0 = DISABLE 1 = ENABLE
8	0x0	CLR_GPIO_RST: clear reset GPIO Controller 0 = DISABLE 1 = ENABLE
7	0x1	CLR_UARTB_RST: clear reset UARTB/VFIR Controller 0 = DISABLE 1 = ENABLE
6	0x1	CLR_UARTA_RST: clear reset UARTA Controller 0 = DISABLE 1 = ENABLE
5	0x0	CLR_TMR_RST: clear reset Timer Controller 0 = DISABLE 1 = ENABLE
1	0x0	CLR_COP_RST: clear reset COP. 0 = DISABLE 1 = ENABLE
0	0x1	CLR_CPU_RST: clear reset CPU. 0 = DISABLE 1 = ENABLE

6.5.91 CLK_RST_CONTROLLER_RST_DEV_H_SET_0

Offset: 308h | Read/Write: R/W | Reset: 0b1111111x111111111111x0110x1xx111

Bit	Reset	Description
31	0x1	SET_BSEV_RST: set reset BSEV controller. 0 = DISABLE 1 = ENABLE
30	0x1	SET_BSEA_RST: set reset BSEA controller. 0 = DISABLE 1 = ENABLE
29	0x1	SET_VDE_RST: set reset VDE controller. 0 = DISABLE 1 = ENABLE
28	0x1	SET_MPE_RST: set reset MPE controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
27	0x1	SET_USB3_RST: set reset USB3 controller. 0 = DISABLE 1 = ENABLE
26	0x1	SET_USB2_RST: set reset USB2 controller. 0 = DISABLE 1 = ENABLE
25	0x1	SET_EMC_RST: set reset EMC controller. 0 = DISABLE 1 = ENABLE
23	0x1	SET_UARTC_RST: set reset UARTC Controller 0 = DISABLE 1 = ENABLE
22	0x1	SET_I2C2_RST: set reset I2C 2 controller. 0 = DISABLE 1 = ENABLE
21	0x1	SET_TVDAC_RST: set reset TVDAC controller. 0 = DISABLE 1 = ENABLE
20	0x1	SET_CSI_RST: set reset CSI controller. 0 = DISABLE 1 = ENABLE
19	0x1	SET_HDMI_RST: set reset HDMI 0 = DISABLE 1 = ENABLE
18	0x1	SET_MIPI_RST: set reset MIPI base-band HSIcontroller. 0 = DISABLE 1 = ENABLE
17	0x1	SET_TVO_RST: set reset TVO/CVE controller 0 = DISABLE 1 = ENABLE
16	0x1	SET_DSI_RST: set reset DSI controller 0 = DISABLE 1 = ENABLE
15	0x1	SET_DVC_I2C_RST: set reset DVC-I2C Controller 0 = DISABLE 1 = ENABLE
14	0x1	SET_SBC3_RST: set reset SBC 3 Controller. 0 = DISABLE 1 = ENABLE
12	0x1	SET_SBC2_RST: set reset SBC 2 Controller. 0 = DISABLE 1 = ENABLE
10	0x0	SET_SNOR_RST: set reset NOR Flash Controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
9	0x1	SET_SBC1_RST: set reset SBC 1 Controller. 0 = DISABLE 1 = ENABLE
8	0x1	SET_KFUSE_RST: set reset KFuse controller. 0 = DISABLE 1 = ENABLE
5	0x1	SET_STAT_MON_RST: set reset statistic monitor 0 = DISABLE 1 = ENABLE
2	0x1	SET_APBDMA_RST: set reset APB-DMA. 0 = DISABLE 1 = ENABLE
1	0x1	SET_AHBDMA_RST: set reset AHB-DMA. 0 = DISABLE 1 = ENABLE
0	0x1	SET_MEM_RST: set reset MC. 0 = DISABLE 1 = ENABLE

6.5.92 CLK_RST_CONTROLLER_RST_DEV_H_CLR_0

Offset: 30ch | Read/Write: R/W | Reset: 0b1111111x111111111111x0110x1xx111

Bit	Reset	Description
31	0x1	CLR_BSEV_RST: clear reset BSEV controller. 0 = DISABLE 1 = ENABLE
30	0x1	CLR_BSEA_RST: clear reset BSEA controller. 0 = DISABLE 1 = ENABLE
29	0x1	CLR_VDE_RST: clear reset VDE controller. 0 = DISABLE 1 = ENABLE
28	0x1	CLR_MPE_RST: clear reset MPE controller. 0 = DISABLE 1 = ENABLE
27	0x1	CLR_USB3_RST: clear reset USB3 controller. 0 = DISABLE 1 = ENABLE
26	0x1	CLR_USB2_RST: clear reset USB2 controller. 0 = DISABLE 1 = ENABLE
25	0x1	CLR_EMC_RST: clear reset EMC controller. 0 = DISABLE 1 = ENABLE
23	0x1	CLR_UARTC_RST: clear reset UARTC Controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	0x1	CLR_I2C2_RST: clear reset I2C 2 controller. 0 = DISABLE 1 = ENABLE
21	0x1	CLR_TVDAC_RST: clear reset TVDAC controller. 0 = DISABLE 1 = ENABLE
20	0x1	CLR_CSI_RST: clear reset CSI controller. 0 = DISABLE 1 = ENABLE
19	0x1	CLR_HDMI_RST: clear reset HDMI 0 = DISABLE 1 = ENABLE
18	0x1	CLR_MIPI_RST: clear reset MIPI base-band HSI controller. 0 = DISABLE 1 = ENABLE
17	0x1	CLR_TVO_RST: clear reset TVO/CVE controller 0 = DISABLE 1 = ENABLE
16	0x1	CLR_DSI_RST: clear reset DSI controller 0 = DISABLE 1 = ENABLE
15	0x1	CLR_DVC_I2C_RST: clear reset DVC-I2C Controller 0 = DISABLE 1 = ENABLE
14	0x1	CLR_SBC3_RST: clear reset SBC 3 Controller. 0 = DISABLE 1 = ENABLE
12	0x1	CLR_SBC2_RST: clear reset SBC 2 Controller. 0 = DISABLE 1 = ENABLE
10	0x0	CLR_SNOR_RST: clear reset NOR Flash Controller. 0 = DISABLE 1 = ENABLE
9	0x1	CLR_SBC1_RST: clear reset SBC 1 Controller. 0 = DISABLE 1 = ENABLE
8	0x1	CLR_KFUSE_RST: clear reset KFuse controller. 0 = DISABLE 1 = ENABLE
5	0x1	CLR_STAT_MON_RST: clear reset statistic monitor 0 = DISABLE 1 = ENABLE
2	0x1	CLR_APBDMA_RST: clear reset APB-DMA. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x1	CLR_AHBDMA_RST: clear reset AHB-DMA. 0 = DISABLE 1 = ENABLE
0	0x1	CLR_MEM_RST: clear reset MC. 0 = DISABLE 1 = ENABLE

6.5.93 CLK_RST_CONTROLLER_RST_DEV_U_SET_0

Offset: 310h | Read/Write: R/W | Reset: 0b1111xx1010111111111

Bit	Reset	Description
18	0x1	SET_DSIB_RST: set reset DSIB Controller 0 = DISABLE 1 = ENABLE
17	0x1	SET_I2C_SLOW_RST: set reset I2C_SLOW 0 = DISABLE 1 = ENABLE
16	0x1	SET_NAND_SPEED_RST: set reset NAND_SPEED 0 = DISABLE 1 = ENABLE
15	0x1	SET_DTV_RST: set reset DTV Controller 0 = DISABLE 1 = ENABLE
11	0x0	SET_AVPUCQ_RST: set reset AVPUCQ logic. 0 = DISABLE 1 = ENABLE
10	0x1	SET_PCIEXCLK_RST: set reset PCIEXCLK logic. 0 = DISABLE 1 = ENABLE
9	0x0	SET_CSITE_RST: set reset CSITE controller. 0 = DISABLE 1 = ENABLE
8	0x1	SET_AFI_RST: set reset AFI controller. 0 = DISABLE 1 = ENABLE
7	0x1	SET_OWR_RST: set reset OWR controller. 0 = DISABLE 1 = ENABLE
6	0x1	SET_PCIE_RST: set reset PCIE controller. 0 = DISABLE 1 = ENABLE
5	0x1	SET_SDMMC3_RST: set reset SDMMC3 controller. 0 = DISABLE 1 = ENABLE
4	0x1	SET_SBC4_RST: set reset SBC4 controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3	0x1	SET_I2C3_RST: set reset I2C3 controller. 0 = DISABLE 1 = ENABLE
2	0x1	SET_UARTE_RST: set reset UARTE controller. 0 = DISABLE 1 = ENABLE
1	0x1	SET_UARTD_RST: set reset UARTD controller. 0 = DISABLE 1 = ENABLE
0	0x1	SET_SPEEDO_RST: set reset SPEEDO controller. 0 = DISABLE 1 = ENABLE

6.5.94 CLK_RST_CONTROLLER_RST_DEV_U_CLR_0

Offset: 314h | Read/Write: R/W | Reset: 0b1111xx1010111111111

Bit	Reset	Description
18	0x1	CLR_DSIB_RST: clear reset DSIB Controller 0 = DISABLE 1 = ENABLE
17	0x1	CLR_I2C_SLOW_RST: clear reset I2C_SLOW 0 = DISABLE 1 = ENABLE
16	0x1	CLR_NAND_SPEED_RST: clear reset NAND_SPEED 0 = DISABLE 1 = ENABLE
15	0x1	CLR_DTV_RST: clear reset DTV Controller 0 = DISABLE 1 = ENABLE
11	0x0	CLR_AVPUCQ_RST: clear reset AVPUCQ logic. 0 = DISABLE 1 = ENABLE
10	0x1	CLR_PCIEXCLK_RST: clear reset PCIEXCLK logic. 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CSITE_RST: clear reset CSITE controller. 0 = DISABLE 1 = ENABLE
8	0x1	CLR_AFI_RST: clear reset AFI controller. 0 = DISABLE 1 = ENABLE
7	0x1	CLR_OWR_RST: clear reset OWR controller. 0 = DISABLE 1 = ENABLE
6	0x1	CLR_PCIE_RST: clear reset PCIE controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x1	CLR_SDMMC3_RST: clear reset SDMMC3 controller. 0 = DISABLE 1 = ENABLE
4	0x1	CLR_SBC4_RST: clear reset SBC4 controller. 0 = DISABLE 1 = ENABLE
3	0x1	CLR_I2C3_RST: clear reset I2C3 controller. 0 = DISABLE 1 = ENABLE
2	0x1	CLR_UARTE_RST: clear reset UARTE controller. 0 = DISABLE 1 = ENABLE
1	0x1	CLR_UARTD_RST: clear reset UARTD controller. 0 = DISABLE 1 = ENABLE
0	0x1	CLR_SPEEDO_RST: clear reset SPEEDO controller. 0 = DISABLE 1 = ENABLE

6.5.95 CLK_RST_CONTROLLER_CLK_ENB_L_SET_0

Offset: 320h | Read/Write: R/W | Reset: 0b100000x00000000000000010011xxx0

Bit	Reset	Description
31	0x1	SET_CLK_ENB_CACHE2: set enable clock to COP cache controller. 0 = DISABLE 1 = ENABLE
30	0x0	SET_CLK_ENB_I2S0: set enable clock to I2S0 Controller 0 = DISABLE 1 = ENABLE
29	0x0	SET_CLK_ENB_VCP: set enable clock to vector co-processor. 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_HOST1X: set enable clock to HOST1X. 0 = DISABLE 1 = ENABLE
27	0x0	SET_CLK_ENB_DISP1: set enable clock to DISP1 controller. 0 = DISABLE 1 = ENABLE
26	0x0	SET_CLK_ENB_DISP2: set enable clock to DISP2 controller. 0 = DISABLE 1 = ENABLE
24	0x0	SET_CLK_ENB_3D: set enable clock to 3D controller. 0 = DISABLE 1 = ENABLE
23	0x0	SET_CLK_ENB_ISP: set enable clock to ISP controller. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
22	0x0	SET_CLK_ENB_USBD: set enable clock to USB controller 0 = DISABLE 1 = ENABLE
21	0x0	SET_CLK_ENB_2D: set enable clock to 2D graphics engine. 0 = DISABLE 1 = ENABLE
20	0x0	SET_CLK_ENB_VI: set enable clock to VI controller. 0 = DISABLE 1 = ENABLE
19	0x0	SET_CLK_ENB_EPP: set enable clock to EPP controller. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CLK_ENB_I2S2: set enable clock to I2S 2 controller 0 = DISABLE 1 = ENABLE
17	0x0	SET_CLK_ENB_PWM: set enable clock to PWM (Pulse Width Modulator) 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_SDMMC4: set enable clock to SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	0x0	SET_CLK_ENB_SDMMC1: set enable clock to SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	0x0	SET_CLK_ENB_NDFLASH: set enable clock to NAND flash controller. 0 = DISABLE 1 = ENABLE
12	0x0	SET_CLK_ENB_I2C1: set enable clock to I2C1 Controller 0 = DISABLE 1 = ENABLE
11	0x0	SET_CLK_ENB_I2S1: set enable clock to I2S1 Controller 0 = DISABLE 1 = ENABLE
10	0x0	SET_CLK_ENB_SPDIF: set enable clock to SPDIF Controller 0 = DISABLE 1 = ENABLE
9	0x0	SET_CLK_ENB_SDMMC2: set enable clock to SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	0x1	SET_CLK_ENB_GPIO: set enable clock to GPIO Controller 0 = DISABLE 1 = ENABLE
7	0x0	SET_CLK_ENB_UARTB: set enable clock to UARTB/VFIR Controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x0	SET_CLK_ENB_UARTA: set enable clock to UARTA Controller 0 = DISABLE 1 = ENABLE
5	0x1	SET_CLK_ENB_TMR: set enable clock to Timer Controller 0 = DISABLE 1 = ENABLE
4	0x1	SET_CLK_ENB_RTC: set enable clock to RTC Controller 0 = DISABLE 1 = ENABLE
0	0x0	SET_CLK_ENB_CPU: set enable clock to CPU. [CCLK Multi-Address 0 = DISABLE 1 = ENABLE

6.5.96 CLK_RST_CONTROLLER_CLK_ENB_L_CLR_0

Offset: 324h | Read/Write: R/W | Reset: 0b100000x00000000000000010011xxx0

Bit	Reset	Description
31	0x1	CLR_CLK_ENB_CACHE2: clear enable clock to COP cache controller. 0 = DISABLE 1 = ENABLE
30	0x0	CLR_CLK_ENB_I2S0: clear enable clock to I2S0 Controller 0 = DISABLE 1 = ENABLE
29	0x0	CLR_CLK_ENB_VCP: clear enable clock to vector co-processor. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_HOST1X: clear enable clock to HOST1X. 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_DISP1: clear enable clock to DISP1 controller. 0 = DISABLE 1 = ENABLE
26	0x0	CLR_CLK_ENB_DISP2: clear enable clock to DISP2 controller. 0 = DISABLE 1 = ENABLE
24	0x0	CLR_CLK_ENB_3D: clear enable clock to 3D controller. 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CLK_ENB_ISP: clear enable clock to ISP controller. 0 = DISABLE 1 = ENABLE
22	0x0	CLR_CLK_ENB_USBD: clear enable clock to USB controller 0 = DISABLE 1 = ENABLE
21	0x0	CLR_CLK_ENB_2D: clear enable clock to 2D graphics engine. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20	0x0	CLR_CLK_ENB_VI: clear enable clock to VI controller. 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CLK_ENB_EPP: clear enable clock to EPP controller. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_I2S2: clear enable clock to I2S 2 controller 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CLK_ENB_PWM: clear enable clock to PWM (Pulse Width Modulator) 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_SDMMC4: clear enable clock to SDMMC4 controller. 0 = DISABLE 1 = ENABLE
14	0x0	CLR_CLK_ENB_SDMMC1: clear enable clock to SDMMC1 controller. 0 = DISABLE 1 = ENABLE
13	0x0	CLR_CLK_ENB_NDFLASH: clear enable clock to NAND flash controller. 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_I2C1: clear enable clock to I2C1 Controller 0 = DISABLE 1 = ENABLE
11	0x0	CLR_CLK_ENB_I2S1: clear enable clock to I2S1 Controller 0 = DISABLE 1 = ENABLE
10	0x0	CLR_CLK_ENB_SPDIF: clear enable clock to SPDIF Controller 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CLK_ENB_SDMMC2: clear enable clock to SDMMC2 controller. 0 = DISABLE 1 = ENABLE
8	0x1	CLR_CLK_ENB_GPIO: clear enable clock to GPIO Controller 0 = DISABLE 1 = ENABLE
7	0x0	CLR_CLK_ENB_UARTB: clear enable clock to UARTB/VFIR Controller 0 = DISABLE 1 = ENABLE
6	0x0	CLR_CLK_ENB_UARTA: clear enable clock to UARTA Controller 0 = DISABLE 1 = ENABLE
5	0x1	CLR_CLK_ENB_TMR: clear enable clock to Timer Controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x1	CLR_CLK_ENB_RTC: clear enable clock to RTC Controller 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CLK_ENB_CPU: clear enable clock to CPU. 0 = DISABLE 1 = ENABLE

6.5.97 CLK_RST_CONTROLLER_CLK_ENB_H_SET_0

Offset: 328h | Read/Write: R/W | Reset: 0b0000000x000000000000x1001000x000

Bit	Reset	Description
31	0x0	SET_CLK_ENB_BSEV: set enable clock to BSEV Controller. 0 = DISABLE 1 = ENABLE
30	0x0	SET_CLK_ENB_BSEA: set enable clock to BSEA Controller 0 = DISABLE 1 = ENABLE
29	0x0	SET_CLK_ENB_VDE: set enable clock to VDE Controller 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_MPE: set enable clock to MPE controller. 0 = DISABLE 1 = ENABLE
27	0x0	SET_CLK_ENB_USB3: set enable clock to USB3 controller. 0 = DISABLE 1 = ENABLE
26	0x0	SET_CLK_ENB_USB2: set enable clock to USB2 controller. 0 = DISABLE 1 = ENABLE
25	0x0	SET_CLK_ENB_EMC: set enable clock to MC/EMC controller. 0 = DISABLE 1 = ENABLE
23	0x0	SET_CLK_ENB_UARTC: set enable clock to UARTC Controller 0 = DISABLE 1 = ENABLE
22	0x0	SET_CLK_ENB_I2C2: set enable clock to I2C2 controller. 0 = DISABLE 1 = ENABLE
21	0x0	SET_CLK_ENB_TVDAC: set enable clock to TVDAC controller. 0 = DISABLE 1 = ENABLE
20	0x0	SET_CLK_ENB_CSI: set enable clock to CSI controller 0 = DISABLE 1 = ENABLE
19	0x0	SET_CLK_ENB_HDMI: set enable clock to HDMI 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	0x0	SET_CLK_ENB_MIPI: set enable clock to MIPI base-band HSI controller 0 = DISABLE 1 = ENABLE
17	0x0	SET_CLK_ENB_TVO: set enable clock to TVO/CVE controller 0 = DISABLE 1 = ENABLE
16	0x0	SET_CLK_ENB_DSI: set enable clock to DSI controller 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_DVC_I2C: set enable clock to DVC-I2C Controller 0 = DISABLE 1 = ENABLE
14	0x0	SET_CLK_ENB_SBC3: set enable clock to SBC 3 (SPI 3) Controller. 0 = DISABLE 1 = ENABLE
12	0x0	SET_CLK_ENB_SBC2: set enable clock to SBC 2 (SPI 2) Controller. 0 = DISABLE 1 = ENABLE
10	0x1	SET_CLK_ENB_SNOR: set enable clock to NOR Flash Controller. 0 = DISABLE 1 = ENABLE
9	0x0	SET_CLK_ENB_SBC1: set enable clock to SBC 1 Controller. 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_KFUSE: set enable clock to KFUSE controller. 0 = DISABLE 1 = ENABLE
7	0x0	SET_CLK_ENB_FUSE: set enable clock to FUSE controller. 0 = DISABLE 1 = ENABLE
6	0x0	SET_CLK_ENB_PMC: set enable clock to PMC controller. 0 = DISABLE 1 = ENABLE
5	0x0	SET_CLK_ENB_STAT_MON: set enable clock to statistic monitor. 0 = DISABLE 1 = ENABLE
4	0x0	SET_CLK_ENB_KBC: set enable clock to keyboard controller. 0 = DISABLE 1 = ENABLE
2	0x0	SET_CLK_ENB_APBDMA: set enable clock to APB-DMA. 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_AHBDMA: set enable clock to AHB-DMA. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	SET_CLK_ENB_MEM: set enable clock to MC/EMC. 0 = DISABLE 1 = ENABLE

6.5.98 CLK_RST_CONTROLLER_CLK_ENB_H_CLR_0

Offset: 32ch | Read/Write: R/W | Reset: 0b0000000x000000000000x1001000x000

Bit	Reset	Description
31	0x0	CLR_CLK_ENB_BSEV: clear enable clock to BSEV Controller. 0 = DISABLE 1 = ENABLE
30	0x0	CLR_CLK_ENB_BSEA: clear enable clock to BSEA Controller 0 = DISABLE 1 = ENABLE
29	0x0	CLR_CLK_ENB_VDE: clear enable clock to VDE Controller 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_MPE: clear enable clock to MPE controller. 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_USB3: clear enable clock to USB3 controller. 0 = DISABLE 1 = ENABLE
26	0x0	CLR_CLK_ENB_USB2: clear enable clock to USB2 controller. 0 = DISABLE 1 = ENABLE
25	0x0	CLR_CLK_ENB_EMC: clear enable clock to MC/EMC controller. 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CLK_ENB_UARTC: clear enable clock to UARTC Controller 0 = DISABLE 1 = ENABLE
22	0x0	CLR_CLK_ENB_I2C2: clear enable clock to I2C2 controller. 0 = DISABLE 1 = ENABLE
21	0x0	CLR_CLK_ENB_TVDAC: clear enable clock to TVDAC controller. 0 = DISABLE 1 = ENABLE
20	0x0	CLR_CLK_ENB_CSI: clear enable clock to CSI controller 0 = DISABLE 1 = ENABLE
19	0x0	CLR_CLK_ENB_HDMI: clear enable clock to HDMI 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_MIPI: clear enable clock to MIPI base-band HSI controller 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
17	0x0	CLR_CLK_ENB_TVO: clear enable clock to TVO/CVE controller 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CLK_ENB_DSI: clear enable clock to DSI controller 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_DVC_I2C: clear enable clock to DVC-I2C Controller 0 = DISABLE 1 = ENABLE
14	0x0	CLR_CLK_ENB_SBC3: clear enable clock to SBC 3 Controller. 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_SBC2: clear enable clock to SBC 2 Controller. 0 = DISABLE 1 = ENABLE
10	0x1	CLR_CLK_ENB_SNOR: clear enable clock to NOR Flash Controller. 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CLK_ENB_SBC1: clear enable clock to SBC 1 Controller. 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CLK_ENB_KFUSE: clear enable clock to KFUSE controller. 0 = DISABLE 1 = ENABLE
7	0x0	CLR_CLK_ENB_FUSE: clear enable clock to FUSE controller. 0 = DISABLE 1 = ENABLE
6	0x0	CLR_CLK_ENB_PMC: clear enable clock to PMC controller. 0 = DISABLE 1 = ENABLE
5	0x0	CLR_CLK_ENB_STAT_MON: clear enable clock to statistic monitor. 0 = DISABLE 1 = ENABLE
4	0x0	CLR_CLK_ENB_KBC: clear enable clock to keyboard controller. 0 = DISABLE 1 = ENABLE
2	0x0	CLR_CLK_ENB_APBDMA: clear enable clock to APB-DMA. 0 = DISABLE 1 = ENABLE
1	0x0	CLR_CLK_ENB_AHBDMA: clear enable clock to AHB-DMA. 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CLK_ENB_MEM: clear enable clock to MC 0 = DISABLE 1 = ENABLE

6.5.99 CLK_RST_CONTROLLER_CLK_ENB_U_SET_0

Offset: 330h | Read/Write: R/W | Reset: 0b0x1x11111x0000xx01x1000000000

Bit	Reset	Description
28	0x0	SET_CLK_ENB_SUS_OUT: set enable clock to SUS pad. 0 = DISABLE 1 = ENABLE
26	0x1	SET_CLK_M_DOUBLER_ENB: set enable CLK_M clk doubler. 0 = DISABLE 1 = ENABLE
25	0x1	SET_SYNC_CLK_DOUBLER_ENB: set enable audio sync clk doubler. 0 = DISABLE 1 = ENABLE
24	0x1	SET_CLK_ENB_CRAM2: set enable COP cache RAM clk. 0 = DISABLE 1 = ENABLE
23	0x1	SET_CLK_ENB_IRAMD: set enable IRAMD clk. 0 = DISABLE 1 = ENABLE
22	0x1	SET_CLK_ENB_IRAMC: set enable IRAMC clk. 0 = DISABLE 1 = ENABLE
21	0x1	SET_CLK_ENB_IRAMB: set enable IRAMB clk. 0 = DISABLE 1 = ENABLE
20	0x1	SET_CLK_ENB_IRAMA: set enable IRAMB clk. 0 = DISABLE 1 = ENABLE
18	0x0	SET_CLK_ENB_DSIB: set enable clock to DSIB Controller 0 = DISABLE 1 = ENABLE
17	0x0	SET_CLK_ENB_I2C_SLOW: set enable clock to I2C_SLOW 0 = DISABLE 1 = ENABLE
16	0x0	SET_CLK_ENB_NAND_SPEED: set enable clock to NAND_SPEED 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_DTV: set enable clock to DTV Controller. Moved to U:15 0 = DISABLE 1 = ENABLE
11	0x1	SET_CLK_ENB_AVPUCQ: set enable clock to AVPUCQ. 0 = DISABLE 1 = ENABLE
9	0x1	SET_CLK_ENB_CSITE: set enable clock to CSITE. 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_AFI: set enable clock to AFI. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
7	0x0	SET_CLK_ENB_OWR: set enable clock to OWR. 0 = DISABLE 1 = ENABLE
6	0x0	SET_CLK_ENB_PCIE: set enable clock to PCIE. 0 = DISABLE 1 = ENABLE
5	0x0	SET_CLK_ENB_SDMMC3: set enable clock to SDMMC3. 0 = DISABLE 1 = ENABLE
4	0x0	SET_CLK_ENB_SBC4: set enable clock to SBC4. 0 = DISABLE 1 = ENABLE
3	0x0	SET_CLK_ENB_I2C3: set enable clock to I2C3. 0 = DISABLE 1 = ENABLE
2	0x0	SET_CLK_ENB_UARTE: set enable clock to UARTE. 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_UARTD: set enable clock to UARTD. 0 = DISABLE 1 = ENABLE
0	0x0	SET_CLK_ENB_SPEEDO: set enable clock to SPEEDO. 0 = DISABLE 1 = ENABLE

6.5.100 CLK_RST_CONTROLLER_CLK_ENB_U_CLR_0

Offset: 334h | Read/Write: R/W | Reset: 0b0x1x11111x0000xx01x1000000000

Bit	Reset	Description
28	0x0	CLR_CLK_ENB_SUS_OUT: clear enable clock to SUS pad. 0 = DISABLE 1 = ENABLE
26	0x1	CLR_CLK_M_DOUBLER_ENB: clear enable CLK_M clk doubler. 0 = DISABLE 1 = ENABLE
25	0x1	CLR_SYNC_CLK_DOUBLER_ENB: clear enable audio sync clk doubler. 0 = DISABLE 1 = ENABLE
24	0x1	CLR_CLK_ENB_CRAM2: clear enable COP cache RAM clk. 0 = DISABLE 1 = ENABLE
23	0x1	CLR_CLK_ENB_IRAMD: clear enable IRAMD clk. 0 = DISABLE 1 = ENABLE
22	0x1	CLR_CLK_ENB_IRAMC: clear enable IRAMC clk. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
21	0x1	CLR_CLK_ENB_IRAMB: clear enable IRAMB clk. 0 = DISABLE 1 = ENABLE
20	0x1	CLR_CLK_ENB_IRAMA: clear enable IRAMB clk. 0 = DISABLE 1 = ENABLE
18	0x0	CLR_CLK_ENB_DSIB: clear enable clock to DSIB Controller 0 = DISABLE 1 = ENABLE
17	0x0	CLR_CLK_ENB_I2C_SLOW: clear enable clock to I2C_SLOW 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CLK_ENB_NAND_SPEED: clear enable clock to NAND_SPEED 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_DTV: clear enable clock to DTV Controller 0 = DISABLE 1 = ENABLE
11	0x1	CLR_CLK_ENB_AVPUCQ: clear enable clock to AVPUCQ. 0 = DISABLE 1 = ENABLE
9	0x1	CLR_CLK_ENB_CSITE: clear enable clock to CSITE. 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CLK_ENB_AFI: clear enable clock to AFI. 0 = DISABLE 1 = ENABLE
7	0x0	CLR_CLK_ENB_OWR: clear enable clock to OWR. 0 = DISABLE 1 = ENABLE
6	0x0	CLR_CLK_ENB_PCIE: clear enable clock to PCIE. 0 = DISABLE 1 = ENABLE
5	0x0	CLR_CLK_ENB_SDMMC3: clear enable clock to SDMMC3. 0 = DISABLE 1 = ENABLE
4	0x0	CLR_CLK_ENB_SBC4: clear enable clock to SBC4. 0 = DISABLE 1 = ENABLE
3	0x0	CLR_CLK_ENB_I2C3: clear enable clock to I2C3. 0 = DISABLE 1 = ENABLE
2	0x0	CLR_CLK_ENB_UARTE: clear enable clock to UARTE. 0 = DISABLE 1 = ENABLE
1	0x0	CLR_CLK_ENB_UARTD: clear enable clock to UARTD. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	CLR_CLK_ENB_SPEEDO: clear enable clock to SPEEDO. 0 = DISABLE 1 = ENABLE

6.5.101 CLK_RST_CONTROLLER_RST_CPU_CMPLX_SET_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 340h | Read/Write: R/W | Reset: 0b000xxxxxxxxxxx1110111011101110

Bit	Reset	Description
30	0x0	SET_PRESETDBG: 1 = assert nPRESETDBG to the coresight. 0 = DISABLE 1 = ENABLE
29	0x0	SET_SCURESET: 1 = assert nSCURESET to the SCU. 0 = DISABLE 1 = ENABLE
28	0x0	SET_PERIPHRESET: 1 = assert nPERIPHRESET to the CPU. 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	0x0	SET_DBGRESET0: 1 = assert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
11	0x1	SET_WDRESET3: 1 = assert nWDRESET to CPU3. 0 = DISABLE 1 = ENABLE
10	0x1	SET_WDRESET2: 1 = assert nWDRESET to CPU2. 0 = DISABLE 1 = ENABLE
9	0x1	SET_WDRESET1: 1 = assert nWDRESET to CPU1. 0 = DISABLE 1 = ENABLE
8	0x0	SET_WDRESET0: 1 = assert nWDRESET to CPU0. 0 = DISABLE 1 = ENABLE
7	0x1	SET_DERESET3: 1 = assert nDERESET to CPU3. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x1	SET_DERESET2: 1 = assert nDERESET to CPU2. 0 = DISABLE 1 = ENABLE
5	0x1	SET_DERESET1: 1 = assert nDERESET to CPU1. 0 = DISABLE 1 = ENABLE
4	0x0	SET_DERESET0: 1 = assert nDERESET to CPU0. 0 = DISABLE 1 = ENABLE
3	0x1	SET_CPURESET3: 1 = assert nCPURESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	SET_CPURESET2: 1 = assert nCPURESET to CPU2. 0 = DISABLE 1 = ENABLE
1	0x1	SET_CPURESET1: 1 = assert nCPURESET to CPU1. 0 = DISABLE 1 = ENABLE
0	0x0	SET_CPURESET0: 1 = assert nCPURESET to CPU0. 0 = DISABLE 1 = ENABLE

6.5.102 CLK_RST_CONTROLLER_RST_CPU_CMPLX_CLR_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 344h | Read/Write: R/W | Reset: 0b000xxxxxxxxxxx1110111011101110

Bit	Reset	Description
30	0x0	CLR_PRESETDBG: 1 = deassert nPRESETDBG to the coresight. 0 = DISABLE 1 = ENABLE
29	0x0	CLR_SCURESET: 1 = deassert nSCURESET to the SCU. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_PERIPHRESET: 1 = deassert nPERIPHRESET to the CPU's interrupt/timer. 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
13	0x1	CLR_DBGRESET1: 1 = deassert nDBGRESET to CPU1. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	0x0	CLR_DBGRESET0: 1 = deassert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
11	0x1	CLR_WDRESET3: 1 = deassert nWDRESET to CPU3. 0 = DISABLE 1 = ENABLE
10	0x1	CLR_WDRESET2: 1 = deassert nWDRESET to CPU2. 0 = DISABLE 1 = ENABLE
9	0x1	CLR_WDRESET1: 1 = deassert nWDRESET to CPU1. 0 = DISABLE 1 = ENABLE
8	0x0	CLR_WDRESET0: 1 = deassert nWDRESET to CPU0. 0 = DISABLE 1 = ENABLE
7	0x1	CLR_DERESET3: 1 = deassert nDERESET to CPU3. 0 = DISABLE 1 = ENABLE
6	0x1	CLR_DERESET2: 1 = deassert nDERESET to CPU2. 0 = DISABLE 1 = ENABLE
5	0x1	CLR_DERESET1: 1 = deassert nDERESET to CPU1. 0 = DISABLE 1 = ENABLE
4	0x0	CLR_DERESET0: 1 = deassert nDERESET to CPU0. 0 = DISABLE 1 = ENABLE
3	0x1	CLR_CPURESET3: 1 = deassert nCPURESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	CLR_CPURESET2: 1 = deassert nCPURESET to CPU2. 0 = DISABLE 1 = ENABLE
1	0x1	CLR_CPURESET1: 1 = deassert nCPURESET to CPU1. 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CPURESET0: 1 = deassert nCPURESET to CPU0. 0 = DISABLE 1 = ENABLE

6.5.103 CLK_RST_CONTROLLER_CLK_CPU_CMPLX_SET_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 348h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
11	0x0	SET_CPU3_CLK_STP: 1 = assert CPU3 clock stop 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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

6.5.104 CLK_RST_CONTROLLER_CLK_CPU_CMPLX_CLR_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 34ch | Read/Write: R/W | Reset: 0b0000

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

6.5.105 CLK_RST_CONTROLLER_RST_DEVICES_V_0

Offset: 358h | Read/Write: R/W | Reset: 0b1111111xxxxxx11111111111111111

Bit	Reset	Description
29	ENABLE	SWR_HDA_RST: Reset High Def Audio 0 = DISABLE 1 = ENABLE
28	ENABLE	SWR_SATA_RST: Reset SATA 0 = DISABLE 1 = ENABLE
23	ENABLE	SWR_ACTMON_RST: Reset ACTMON 0 = DISABLE 1 = ENABLE
16	ENABLE	SWR_ATOMICS_RST: Reset ATOMICS 0 = DISABLE 1 = ENABLE
15	ENABLE	SWR_HDA2CODEC_2X_RST: Reset HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	ENABLE	SWR_DAM2_RST: Reset DAM2 0 = DISABLE 1 = ENABLE
13	ENABLE	SWR_DAM1_RST: Reset DAM1 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
12	ENABLE	SWR_DAM0_RST: Reset DAM0 0 = DISABLE 1 = ENABLE
11	ENABLE	SWR_APBIF_RST: Reset APBIF 0 = DISABLE 1 = ENABLE
10	ENABLE	SWR_AUDIO_RST: Reset AUDIO 0 = DISABLE 1 = ENABLE
9	ENABLE	SWR_SBC6_RST: Reset SBC6 0 = DISABLE 1 = ENABLE
8	ENABLE	SWR_SBC5_RST: Reset SBC5 0 = DISABLE 1 = ENABLE
7	ENABLE	SWR_I2C4_RST: Reset I2C4 0 = DISABLE 1 = ENABLE
6	ENABLE	SWR_I2S4_RST: Reset I2S4 0 = DISABLE 1 = ENABLE
5	ENABLE	SWR_I2S3_RST: Reset I2S3 0 = DISABLE 1 = ENABLE
4	ENABLE	SWR_TSENSOR_RST: Reset TSENSOR 0 = DISABLE 1 = ENABLE
3	ENABLE	SWR_MSELECT_RST: Reset MSELECT 0 = DISABLE 1 = ENABLE
2	ENABLE	SWR_3D2_RST: Reset 3D2 controller. 0 = DISABLE 1 = ENABLE
1	ENABLE	SWR_CPULP_RST: Reset CPULP. 0 = DISABLE 1 = ENABLE
0	ENABLE	SWR_CPUG_RST: Reset CPUG. 0 = DISABLE 1 = ENABLE

6.5.106 CLK_RST_CONTROLLER_RST_DEVICES_W_0

IOBIST control has clock enable only.

Offset: 35ch | Read/Write: R/W | Reset: 0b11111111111111

Bit	Reset	Description
13	ENABLE	SWR_RESERVED10_RST: Reserved MIPI IOBIST 0 = DISABLE 1 = ENABLE
12	ENABLE	SWR_RESERVED9_RST: Reserved SATA IOBIST 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	ENABLE	SWR_RESERVED8_RST: Reserved HDMI IOBIST 0 = DISABLE 1 = ENABLE
10	ENABLE	SWR_RESERVED7_RST: Reserved EMC IOBIST 0 = DISABLE 1 = ENABLE
9	ENABLE	SWR_RESERVED6_RST: Reserved PCIE2 IOBIST 0 = DISABLE 1 = ENABLE
8	ENABLE	SWR_CEC_RST: Reset CEC 0 = DISABLE 1 = ENABLE
7	ENABLE	SWR_RESERVED5_RST: Reserved PCIERX5 0 = DISABLE 1 = ENABLE
6	ENABLE	SWR_RESERVED4_RST: Reserved PCIERX4 0 = DISABLE 1 = ENABLE
5	ENABLE	SWR_RESERVED3_RST: Reserved PCIERX3 0 = DISABLE 1 = ENABLE
4	ENABLE	SWR_RESERVED2_RST: Reserved PCIERX2 0 = DISABLE 1 = ENABLE
3	ENABLE	SWR_RESERVED1_RST: Reserved PCIERX1 0 = DISABLE 1 = ENABLE
2	ENABLE	SWR_RESERVED0_RST: Reserved PCIERX0 0 = DISABLE 1 = ENABLE
1	ENABLE	SWR_SATACOLD_RST: Reset SATACOLD 0 = DISABLE 1 = ENABLE
0	ENABLE	SWR_HDA2HDMICODEC_RST: Reset HDA2HDMICODEC (no dedicated module or clock). 0 = DISABLE 1 = ENABLE

6.5.107 CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0

Offset: 360h | Read/Write: R/W | Reset: 0b00000001111110000000000000000000

Bit	Reset	Description
29	DISABLE	CLK_ENB_HDA: Enable clock to High Def 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

Bit	Reset	Description
		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 spdif audio sync clk doubler. 0 = DISABLE 1 = ENABLE
21	ENABLE	CLK_ENB_I2S4_DOUBLER: Enable i2s4 audio sync clk doubler. 0 = DISABLE 1 = ENABLE
20	ENABLE	CLK_ENB_I2S3_DOUBLER: Enable i2s3 audio sync clk doubler. 0 = DISABLE 1 = ENABLE
19	ENABLE	CLK_ENB_I2S2_DOUBLER: Enable i2s2 audio sync clk doubler. 0 = DISABLE 1 = ENABLE
18	ENABLE	CLK_ENB_I2S1_DOUBLER: Enable i2s1 audio sync clk doubler. 0 = DISABLE 1 = ENABLE
17	ENABLE	CLK_ENB_I2S0_DOUBLER: Enable i2s0 audio sync clk doubler. 0 = DISABLE 1 = ENABLE
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

Bit	Reset	Description
		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	DISABLE	CLK_ENB_3D2: Enable clock to 3D2 controller. 0 = DISABLE 1 = ENABLE
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

6.5.108 CLK_RST_CONTROLLER_CLK_OUT_ENB_W_0

IOBIST control has clock enable only.

Offset: 364h | Read/Write: R/W | Reset: 0b00000011111100

Bit	Reset	Description
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
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 CLK_ENB_PCIE is the master control for all PCIERX* clocks as well. 0 = DISABLE 1 = ENABLE
7	ENABLE	CLK_ENB_PCIERX5: Enable clock to PCIERX5. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	ENABLE	CLK_ENB_PCIERX4: Enable clock to PCIERX4 0 = DISABLE 1 = ENABLE
5	ENABLE	CLK_ENB_PCIERX3: Enable clock to PCIERX3 0 = DISABLE 1 = ENABLE
4	ENABLE	CLK_ENB_PCIERX2: Enable clock to PCIERX2 0 = DISABLE 1 = ENABLE
3	ENABLE	CLK_ENB_PCIERX1: Enable clock to PCIERX1 0 = DISABLE 1 = ENABLE
2	ENABLE	CLK_ENB_PCIERX0: Enable clock to PCIERX0 0 = DISABLE 1 = ENABLE
1	DISABLE	CLK_ENB_RESERVE0: Unused clk 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

6.5.109 CLK_RST_CONTROLLER_CCLKG_BURST_POLICY_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 368h | Read/Write: R/W | Reset: 0b00010000xxxxxxx00000000000000000

Bit	R/W	Reset	Description
31:28	RW	0x1	CPU_STATE: 0000=32KHz 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 temp 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 = pllP_out3, 0111 = clk_d, 1xxx = PLLX_out0, 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0

Bit	R/W	Reset	Description
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 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0
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 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0
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 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0

6.5.110 CLK_RST_CONTROLLER_SUPER_CCLKG_DIVIDER_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 36ch | Read/Write: R/W | Reset: 0b0xxx00000000000000000000000000

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
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 denote 0.5x)
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_CDIV_DIVISOR: Actual value = n + 1.

6.5.111 CLK_RST_CONTROLLER_CCLKLP_BURST_POLICY_0

[CCLK Multi-Address]: Refer to details in file header

WARNING! The PLLX_DIV2_BYPASS_LP default is low selecting source (0b1000)= pllx/2.

DO NOT set to 1 when PLLX is at full 1GHZ rate.

PLLX_DIV2_BYPASS_LP should only be set to one if both the following conditions are met:

1. cpulp clock is disable
2. PLLX is programmed at lower rate (<= Freq(PLLX)/2).

Offset: 370h | Read/Write: R/W | Reset: 0b00010000xxxxxxx00000000000000000

Bit	R/W	Reset	Description
31:28	RW	0x1	CPULP_STATE: 0000=32KHz 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 = pllP_out3, 0111 = clk_d, 1000 = PLLX_out0/(2-PLLX_DIV2_BYPASS_LP) 0 = CLKM 1 = PLLC_OUT0 2 = CLKS 3 = PLLM_OUT0 4 = PLLP_OUT0 5 = PLLP_OUT4 6 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0
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 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0
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 = PLLP_OUT3

Bit	R/W	Reset	Description
			7 = CLKD 8 = PLLX_OUT0
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 = PLLP_OUT3 7 = CLKD 8 = PLLX_OUT0

6.5.112 CLK_RST_CONTROLLER_SUPER_CCLKLP_DIVIDER_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 374h | Read/Write: R/W | Reset: 0b0xxx00000000000000000000000000

Bit	Reset	Description
31	DISABLE	SUPER_CDIV_ENB: 0 = disable divider. 0 = DISABLE 1 = ENABLE
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 denote 0.5x)
15:8	0x0	SUPER_CDIV_DIVIDEND: Actual value = n + 1.
7:0	0x0	SUPER_CDIV_DIVISOR: Actual value = n + 1.

6.5.113 CLK_RST_CONTROLLER_CLK_CPUG_CMPLX_0

[CCLK Multi-Address]: Refer to details in file header cpu_divN moved to mselect clock. Programmable divider not available.

CPUG complex consists of the CPUG, L2-cache controller, and a number of bridge devices interfacing CPUG/L2-cache to the rest of the system.

Except the CPUG, L2-cache controller, and bridge logic to external memory which always runs at non-divided-down CPUG frequency, other bridge devices can run at various of programmable divided-down CPUG clock ratios.

Note: However, since the CPUG clock can be selected from 1-of-9 clock sources (please refer to CCLKG_BURST_POLICY register), it's the users responsibility to not select a bridge divide-down CPUG clock ratio that will exceed 1/4 of the "MAX" CPUG frequency supported.

The "MAX" CPUG frequency is 1.1GHz.

Offset: 378h | Read/Write: R/W | Reset: 0b0000xxxxxx00

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.

Bit	Reset	Description
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: Unused but available

6.5.114 CLK_RST_CONTROLLER_CLK_CPULP_CMPLX_0

[CCLK Multi-Address]: Refer to details in file header

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

Note: However, since the CPUG clock can be selected from 1-of-9 clock sources (please refer to CCLKG_BURST_POLICY register), it's the users responsibility to not select a bridge divide-down CPUG clock ratio that will exceed 1/4 of the "MAX" CPUG frequency supported.

The "MAX" CPUG frequency is 1.1GHz.

Offset: 37ch | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
8	0x0	CPULP0_CLK_STP: 1 = CPULP0 clock stop, 0 = CPULP0 clock run.

6.5.115 CLK_RST_CONTROLLER_CPU_SOFTRST_CTRL_0

Offset: 380h | Read/Write: R/W | Reset: 0b0000110000000100

Bit	Reset	Description
15:8	0xc	CPU_CLAMP_WIDTH: CPU clamp release pulse width
7:0	0x4	CPU_SOFTRST_WIDTH: CPU soft reset pulse width in addition to clamp.

6.5.116 CLK_RST_CONTROLLER_CLK_SOURCE_G3D2_0

Offset: 3b0h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
31:30	PLLM_OUT0	G3D2_CLK_SRC: 00 = pllM_out0 01 = pllC_out0 10 = pllP_out0 11 = pllA_out0 0 = PLLM_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = PLLA_OUT0
15:8	0x0	G3D2_IDLE_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x) if all 0's, this idle divisor field will not be use. for non-zero values, when host1x is idle, this field will be use instead of G3D2_CLK_DIVISOR.
7:0	0x0	G3D2_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.117 CLK_RST_CONTROLLER_CLK_SOURCE_MSELECT_0

Offset: 3b4h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	MSELECT_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	MSELECT_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.118 CLK_RST_CONTROLLER_CLK_SOURCE_TSENSOR_0

Offset: 3b8h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_S	TSENSOR_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = clk_m 11 = clk_s 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = CLK_M 3 = CLK_S
7:0	0x0	TSENSOR_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.119 CLK_RST_CONTROLLER_CLK_SOURCE_I2S3_0

Offset: 3bch | Read/Write: R/W | Reset: 0b11x1xxxxxxxxxxxxxxxxxxx00000000

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: Obsolete. Use CLK_ENB_I2S3 to control clock enable
7:0	0x0	I2S3_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.120 CLK_RST_CONTROLLER_CLK_SOURCE_I2S4_0

Offset: 3c0h | Read/Write: R/W | Reset: 0b11x1xxxxxxxxxxxxxxxxxxx00000000

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: Obsolete. Use CLK_ENB_I2S4 to control clock enable
7:0	0x0	I2S4_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.121 CLK_RST_CONTROLLER_CLK_SOURCE_I2C4_0

Offset: 3c4h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
31:30	CLK_M	I2C4_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
15:0	0x0	I2C4_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 1.0x)

6.5.122 CLK_RST_CONTROLLER_CLK_SOURCE_SBC5_0

Offset: 3c8h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SBC5_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	SBC5_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.123 CLK_RST_CONTROLLER_CLK_SOURCE_SBC6_0

Offset: 3cch | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SBC6_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	SBC6_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.124 CLK_RST_CONTROLLER_CLK_SOURCE_AUDIO_0

Offset: 3d0h | Read/Write: R/W | Reset: 0b11x1xxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	AUDIO_CLK_SRC: 00 = pllA_out0 01 = pllC_out0 10 = pllP_out0 11 = clk_m 0 = PLLA_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M
28	0x1	AUDIO_MASTER_CLKEN: Obsolete. Use CLK_ENB_AUDIO to control clock enable
7:0	0x0	AUDIO_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.125 CLK_RST_CONTROLLER_CLK_SOURCE_DAM0_0

Offset: 3d8h | Read/Write: R/W | Reset: 0b11x1xxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	DAM0_CLK_SRC: 00 = pllA_out0 01 = pllC_out0 10 = pllP_out0 11 = clk_m 0 = PLLA_OUT0

Bit	Reset	Description
		1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M
28	0x1	DAM0_MASTER_CLKEN: Obsolete. Use CLK_ENB_DAM0 to control clock enable
7:0	0x0	DAM0_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.126 CLK_RST_CONTROLLER_CLK_SOURCE_DAM1_0

Offset: 3dch | Read/Write: R/W | Reset: 0b11x1xxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	DAM1_CLK_SRC: 00 = pllA_out0 01 = pllC_out0 10 = pllP_out0 11 = clk_m 0 = PLLA_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M
28	0x1	DAM1_MASTER_CLKEN: Obsolete. Use CLK_ENB_DAM1 to control clock enable
7:0	0x0	DAM1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.127 CLK_RST_CONTROLLER_CLK_SOURCE_DAM2_0

Offset: 3e0h | Read/Write: R/W | Reset: 0b11x1xxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	DAM2_CLK_SRC: 00 = pllA_out0 01 = pllC_out0 10 = pllP_out0 11 = clk_m 0 = PLLA_OUT0 1 = PLLC_OUT0 2 = PLLP_OUT0 3 = CLK_M
28	0x1	DAM2_MASTER_CLKEN: Obsolete. Use CLK_ENB_DAM2 to control clock enable
7:0	0x0	DAM2_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.128 CLK_RST_CONTROLLER_CLK_SOURCE_HDA2CODEC_2X_0

Offset: 3e4h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	HDA2CODEC_2X_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	HDA2CODEC_2X_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.129 CLK_RST_CONTROLLER_CLK_SOURCE_ACTMON_0

Offset: 3e8h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	ACTMON_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = clk_s 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = CLK_S

Bit	Reset	Description
		3 = CLK_M
7:0	0x0	ACTMON_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.130 CLK_RST_CONTROLLER_CLK_SOURCE_EXTPERIPH1_0

Offset: 3ech | Read/Write: R/W | Reset: 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 1 = CLK_S 2 = PLLP_OUT0 3 = CLK_M 4 = PLLE_OUT0
7:0	0x0	EXTPERIPH1_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.131 CLK_RST_CONTROLLER_CLK_SOURCE_EXTPERIPH2_0

Offset: 3f0h | Read/Write: R/W | Reset: 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 denote 0.5x)

6.5.132 CLK_RST_CONTROLLER_CLK_SOURCE_EXTPERIPH3_0

Offset: 3f4h | Read/Write: R/W | Reset: 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 denote 0.5x)

6.5.133 CLK_RST_CONTROLLER_CLK_SOURCE_NAND_SPEED_0

Offset: 3f8h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	NAND_SPEED_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = pllM_out0 11 = clk_m 0 = PLLP_OUT0

Bit	Reset	Description
		1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
7:0	0x0	NAND_SPEED_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.134 CLK_RST_CONTROLLER_CLK_SOURCE_I2C_SLOW_0

Offset: 3fch | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	I2C_SLOW_CLK_SRC: 00 = pllP_out0 01 = pllC_out0 10 = clk_s 11 = clk_m 0 = PLLP_OUT0 1 = PLLC_OUT0 2 = CLK_S 3 = CLK_M
7:0	0x0	I2C_SLOW_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.135 CLK_RST_CONTROLLER_CLK_SOURCE_SYS_0

Divider only. All other controls in SCLK_BURST_POLICY, SUPER_SCLK_DIVIDER

Offset: 400h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	SYS_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.136 CLK_RST_CONTROLLER_CLK_SOURCE_SPEEDO_0

Offset: 404h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	SPEEDO_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	SPEEDO_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.137 CLK_RST_CONTROLLER_CLK_SOURCE_SATA_OOB_0

Offset: 420h | Read/Write: R/W | Reset: 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

Bit	Reset	Description
		1 = PLLC_OUT0 2 = PLLM_OUT0 3 = CLK_M
7:0	0x0	SATA_OOB_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.138 CLK_RST_CONTROLLER_CLK_SOURCE_SATA_0

Offset: 424h | Read/Write: R/W | Reset: 0b11xxxxx1xxxxxxxxxxxxxxxxx00000000

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 denote 0.5x)

6.5.139 CLK_RST_CONTROLLER_CLK_SOURCE_HDA_0

Offset: 428h | Read/Write: R/W | Reset: 0b11xxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31:30	CLK_M	HDA_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	HDA_CLK_DIVISOR: N = Divide by (n+1) (lsb denote 0.5x)

6.5.140 CLK_RST_CONTROLLER_RST_DEV_V_SET_0

Offset: 430h | Read/Write: R/W | Reset: 0b1111111xxxxxx11111111111111111

Bit	Reset	Description
29	0x1	SET_HDA_RST: set reset HDA 0 = DISABLE 1 = ENABLE
28	0x1	SET_SATA_RST: set reset SATA 0 = DISABLE 1 = ENABLE
23	0x1	SET_ACTMON_RST: set reset ACTMON 0 = DISABLE 1 = ENABLE
16	0x1	SET_ATOMICS_RST: set reset ATOMICS 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
15	0x1	SET_HDA2CODEC_2X_RST: set reset HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	0x1	SET_DAM2_RST: set reset DAM2 0 = DISABLE 1 = ENABLE
13	0x1	SET_DAM1_RST: set reset DAM1 0 = DISABLE 1 = ENABLE
12	0x1	SET_DAM0_RST: set reset DAM0 0 = DISABLE 1 = ENABLE
11	0x1	SET_APBIF_RST: set reset APBIF 0 = DISABLE 1 = ENABLE
10	0x1	SET_AUDIO_RST: set reset AUDIO 0 = DISABLE 1 = ENABLE
9	0x1	SET_SBC6_RST: set reset SBC6 0 = DISABLE 1 = ENABLE
8	0x1	SET_SBC5_RST: set reset SBC5 0 = DISABLE 1 = ENABLE
7	0x1	SET_I2C4_RST: set reset I2C4 0 = DISABLE 1 = ENABLE
6	0x1	SET_I2S4_RST: set reset I2S4 0 = DISABLE 1 = ENABLE
5	0x1	SET_I2S3_RST: set reset I2S3 0 = DISABLE 1 = ENABLE
4	0x1	SET_TSENSOR_RST: set reset TSENSOR 0 = DISABLE 1 = ENABLE
3	0x1	SET_MSELECT_RST: set reset MSELECT 0 = DISABLE 1 = ENABLE
2	0x1	SET_3D2_RST: set reset 3D2 controller. 0 = DISABLE 1 = ENABLE
1	0x1	SET_CPULP_RST: set reset CPULP. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE
0	0x1	SET_CPUG_RST: set reset CPUG. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE

6.5.141 CLK_RST_CONTROLLER_RST_DEV_V_CLR_0

Offset: 434h | Read/Write: R/W | Reset: 0b1111111xxxxxx11111111111111111

Bit	Reset	Description
29	0x1	CLR_HDA_RST: clr reset HDA 0 = DISABLE 1 = ENABLE
28	0x1	CLR_SATA_RST: clr reset SATA 0 = DISABLE 1 = ENABLE
23	0x1	CLR_ACTMON_RST: clr reset ACTMON 0 = DISABLE 1 = ENABLE
16	0x1	CLR_ATOMICS_RST: clr reset ATOMICS 0 = DISABLE 1 = ENABLE
15	0x1	CLR_HDA2CODEC_2X_RST: clr reset HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	0x1	CLR_DAM2_RST: clr reset DAM2 0 = DISABLE 1 = ENABLE
13	0x1	CLR_DAM1_RST: clr reset DAM1 0 = DISABLE 1 = ENABLE
12	0x1	CLR_DAM0_RST: clr reset DAM0 0 = DISABLE 1 = ENABLE
11	0x1	CLR_APBIF_RST: clr reset APBIF 0 = DISABLE 1 = ENABLE
10	0x1	CLR_AUDIO_RST: clr reset AUDIO 0 = DISABLE 1 = ENABLE
9	0x1	CLR_SBC6_RST: clr reset SBC6 0 = DISABLE 1 = ENABLE
8	0x1	CLR_SBC5_RST: clr reset SBC5 0 = DISABLE 1 = ENABLE
7	0x1	CLR_I2C4_RST: clr reset I2C4 0 = DISABLE 1 = ENABLE
6	0x1	CLR_I2S4_RST: clr reset I2S4 0 = DISABLE 1 = ENABLE
5	0x1	CLR_I2S3_RST: clr reset I2S3 0 = DISABLE 1 = ENABLE
4	0x1	CLR_TSENSOR_RST: clr reset TSENSOR 0 = DISABLE 1 = ENABLE
3	0x1	CLR_MSELECT_RST: clr reset MSELECT 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x1	CLR_3D2_RST: clr reset 3D2 controller. 0 = DISABLE 1 = ENABLE
1	0x1	CLR_CPULP_RST: clr reset CPULP. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE
0	0x1	CLR_CPUG_RST: clr reset CPUG. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE

6.5.142 CLK_RST_CONTROLLER_RST_DEV_W_SET_0

Offset: 438h | Read/Write: R/W | Reset: 0b11111111111111

Bit	Reset	Description
13	0x1	SET_RESERVED10_RST: reserved MIPI IOBIST 0 = DISABLE 1 = ENABLE
12	0x1	SET_RESERVED9_RST: reserved SATA IOBIST 0 = DISABLE 1 = ENABLE
11	0x1	SET_RESERVED8_RST: reserved HDMI IOBIST 0 = DISABLE 1 = ENABLE
10	0x1	SET_RESERVED7_RST: reserved EMC IOBIST 0 = DISABLE 1 = ENABLE
9	0x1	SET_RESERVED6_RST: reserved PCIE2 IOBIST 0 = DISABLE 1 = ENABLE
8	0x1	SET_CEC_RST: set reset CEC 0 = DISABLE 1 = ENABLE
7	0x1	SET_RESERVED5_RST: reserved PCIERX5 0 = DISABLE 1 = ENABLE
6	0x1	SET_RESERVED4_RST: reserved PCIERX4 0 = DISABLE 1 = ENABLE
5	0x1	SET_RESERVED3_RST: reserved PCIERX3 0 = DISABLE 1 = ENABLE
4	0x1	SET_RESERVED2_RST: reserved PCIERX2 0 = DISABLE 1 = ENABLE
3	0x1	SET_RESERVED1_RST: reserved PCIERX1 0 = DISABLE 1 = ENABLE
2	0x1	SET_RESERVED0_RST: reserved PCIERX0 0 = DISABLE 1 = ENABLE
1	0x1	SET_SATACOLD_RST: set reset SATACOLD 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x1	SET_HDA2HDMICODEC_RST: set reset HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

6.5.143 CLK_RST_CONTROLLER_RST_DEV_W_CLR_0

Offset: 43ch | Read/Write: R/W | Reset: 0b11111111111111

Bit	Reset	Description
13	0x1	CLR_RESERVED10_RST: reserved MIPI IOBIST 0 = DISABLE 1 = ENABLE
12	0x1	CLR_RESERVED9_RST: reserved SATA IOBIST 0 = DISABLE 1 = ENABLE
11	0x1	CLR_RESERVED8_RST: reserved HDMI IOBIST 0 = DISABLE 1 = ENABLE
10	0x1	CLR_RESERVED7_RST: reserved EMC IOBIST 0 = DISABLE 1 = ENABLE
9	0x1	CLR_RESERVED6_RST: reserved PCIE2 IOBIST 0 = DISABLE 1 = ENABLE
8	0x1	CLR_CEC_RST: clr reset CEC 0 = DISABLE 1 = ENABLE
7	0x1	CLR_RESERVED5_RST: reserved PCIERX5 0 = DISABLE 1 = ENABLE
6	0x1	CLR_RESERVED4_RST: reserved PCIERX4 0 = DISABLE 1 = ENABLE
5	0x1	CLR_RESERVED3_RST: reserved PCIERX3 0 = DISABLE 1 = ENABLE
4	0x1	CLR_RESERVED2_RST: reserved PCIERX2 0 = DISABLE 1 = ENABLE
3	0x1	CLR_RESERVED1_RST: reserved PCIERX1 0 = DISABLE 1 = ENABLE
2	0x1	CLR_RESERVED0_RST: reserved PCIERX0 0 = DISABLE 1 = ENABLE
1	0x1	CLR_SATACOLD_RST: clr reset SATACOLD 0 = DISABLE 1 = ENABLE
0	0x1	CLR_HDA2HDMICODEC_RST: clr reset HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

6.5.144 CLK_RST_CONTROLLER_CLK_ENB_V_SET_0

Offset: 440h | Read/Write: R/W | Reset: 0b00000001111110000000000000000

Bit	Reset	Description
29	0x0	SET_CLK_ENB_HDA: set enable clock to HDA 0 = DISABLE 1 = ENABLE
28	0x0	SET_CLK_ENB_SATA: set enable clock to SATA 0 = DISABLE 1 = ENABLE
27	0x0	SET_CLK_ENB_SATA_OOB: set enable clock to SATA_OOB 0 = DISABLE 1 = ENABLE
26	0x0	SET_CLK_ENB_EXTPERIPH3: set enable clock to EXTPERIPH3 0 = DISABLE 1 = ENABLE
25	0x0	SET_CLK_ENB_EXTPERIPH2: set enable clock to EXTPERIPH2 0 = DISABLE 1 = ENABLE
24	0x0	SET_CLK_ENB_EXTPERIPH1: set enable clock to EXTPERIPH1 0 = DISABLE 1 = ENABLE
23	0x0	SET_CLK_ENB_ACTMON: set enable clock to ACTMON 0 = DISABLE 1 = ENABLE
22	0x1	SET_CLK_ENB_SPDIF_DOUBLER: set enable clock to SPDIF audio sync doubler 0 = DISABLE 1 = ENABLE
21	0x1	SET_CLK_ENB_I2S4_DOUBLER: set enable clock to I2S4 audio sync doubler 0 = DISABLE 1 = ENABLE
20	0x1	SET_CLK_ENB_I2S3_DOUBLER: set enable clock to I2S3 audio sync doubler 0 = DISABLE 1 = ENABLE
19	0x1	SET_CLK_ENB_I2S2_DOUBLER: set enable clock to I2S2 audio sync doubler 0 = DISABLE 1 = ENABLE
18	0x1	SET_CLK_ENB_I2S1_DOUBLER: set enable clock to I2S1 audio sync doubler 0 = DISABLE 1 = ENABLE
17	0x1	SET_CLK_ENB_I2S0_DOUBLER: set enable clock to I2S0 audio sync doubler 0 = DISABLE 1 = ENABLE
16	0x0	SET_CLK_ENB_ATOMICS: set enable clock to ATOMICS 0 = DISABLE 1 = ENABLE
15	0x0	SET_CLK_ENB_HDA2CODEC_2X: set enable clock to HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	0x0	SET_CLK_ENB_DAM2: set enable clock to DAM2 0 = DISABLE 1 = ENABLE
13	0x0	SET_CLK_ENB_DAM1: set enable clock to DAM1 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	0x0	SET_CLK_ENB_DAM0: set enable clock to DAM0 0 = DISABLE 1 = ENABLE
11	0x0	SET_CLK_ENB_APBIF: set enable clock to APBIF 0 = DISABLE 1 = ENABLE
10	0x0	SET_CLK_ENB_AUDIO: set enable clock to AUDIO 0 = DISABLE 1 = ENABLE
9	0x0	SET_CLK_ENB_SBC6: set enable clock to SBC6 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_SBC5: set enable clock to SBC5 0 = DISABLE 1 = ENABLE
7	0x0	SET_CLK_ENB_I2C4: set enable clock to I2C4 0 = DISABLE 1 = ENABLE
6	0x0	SET_CLK_ENB_I2S4: set enable clock to I2S4 0 = DISABLE 1 = ENABLE
5	0x0	SET_CLK_ENB_I2S3: set enable clock to I2S3 0 = DISABLE 1 = ENABLE
4	0x0	SET_CLK_ENB_TSENSOR: set enable clock to TSENSOR 0 = DISABLE 1 = ENABLE
3	0x0	SET_CLK_ENB_MSELECT: set enable clock to MSELECT 0 = DISABLE 1 = ENABLE
2	0x0	SET_CLK_ENB_3D2: set enable clock to 3D2 controller. 0 = DISABLE 1 = ENABLE
1	0x0	SET_CLK_ENB_CPULP: set enable clock to CPULP. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE
0	0x0	SET_CLK_ENB_CPUG: set enable clock to CPUG. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE

6.5.145 CLK_RST_CONTROLLER_CLK_ENB_V_CLR_0

Offset: 444h | Read/Write: R/W | Reset: 0b00000001111110000000000000000000

Bit	Reset	Description
29	0x0	CLR_CLK_ENB_HDA: clear enable clock to HDA 0 = DISABLE 1 = ENABLE
28	0x0	CLR_CLK_ENB_SATA: clear enable clock to SATA 0 = DISABLE 1 = ENABLE
27	0x0	CLR_CLK_ENB_SATA_OOB: clear enable clock to SATA_OOB 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26	0x0	CLR_CLK_ENB_EXTPERIPH3: clear enable clock to EXTPERIPH3 0 = DISABLE 1 = ENABLE
25	0x0	CLR_CLK_ENB_EXTPERIPH2: clear enable clock to EXTPERIPH2 0 = DISABLE 1 = ENABLE
24	0x0	CLR_CLK_ENB_EXTPERIPH1: clear enable clock to EXTPERIPH1 0 = DISABLE 1 = ENABLE
23	0x0	CLR_CLK_ENB_ACTMON: clear enable clock to ACTMON 0 = DISABLE 1 = ENABLE
22	0x1	CLR_CLK_ENB_SPDIF_DOUBLER: clear enable clock to SPDIF audio sync doubler 0 = DISABLE 1 = ENABLE
21	0x1	CLR_CLK_ENB_I2S4_DOUBLER: clear enable clock to I2S4 audio sync doubler 0 = DISABLE 1 = ENABLE
20	0x1	CLR_CLK_ENB_I2S3_DOUBLER: clear enable clock to I2S3 audio sync doubler 0 = DISABLE 1 = ENABLE
19	0x1	CLR_CLK_ENB_I2S2_DOUBLER: clear enable clock to I2S2 audio sync doubler 0 = DISABLE 1 = ENABLE
18	0x1	CLR_CLK_ENB_I2S1_DOUBLER: clear enable clock to I2S1 audio sync doubler 0 = DISABLE 1 = ENABLE
17	0x1	CLR_CLK_ENB_I2S0_DOUBLER: clear enable clock to I2S0 audio sync doubler 0 = DISABLE 1 = ENABLE
16	0x0	CLR_CLK_ENB_ATOMICS: clear enable clock to ATOMICS 0 = DISABLE 1 = ENABLE
15	0x0	CLR_CLK_ENB_HDA2CODEC_2X: clear enable clock to HDA2CODEC_2X 0 = DISABLE 1 = ENABLE
14	0x0	CLR_CLK_ENB_DAM2: clear enable clock to DAM2 0 = DISABLE 1 = ENABLE
13	0x0	CLR_CLK_ENB_DAM1: clear enable clock to DAM1 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_DAM0: clear enable clock to DAM0 0 = DISABLE 1 = ENABLE
11	0x0	CLR_CLK_ENB_APBIF: clear enable clock to APBIF 0 = DISABLE 1 = ENABLE
10	0x0	CLR_CLK_ENB_AUDIO: clear enable clock to AUDIO 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CLK_ENB_SBC6: clear enable clock to SBC6 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	CLR_CLK_ENB_SBC5: clear enable clock to SBC5 0 = DISABLE 1 = ENABLE
7	0x0	CLR_CLK_ENB_I2C4: clear enable clock to I2C4 0 = DISABLE 1 = ENABLE
6	0x0	CLR_CLK_ENB_I2S4: clear enable clock to I2S4 0 = DISABLE 1 = ENABLE
5	0x0	CLR_CLK_ENB_I2S3: clear enable clock to I2S3 0 = DISABLE 1 = ENABLE
4	0x0	CLR_CLK_ENB_TSENSOR: clear enable clock to TSENSOR 0 = DISABLE 1 = ENABLE
3	0x0	CLR_CLK_ENB_MSELECT: clear enable clock to MSELECT 0 = DISABLE 1 = ENABLE
2	0x0	CLR_CLK_ENB_3D2: clear enable clock to 3D2 controller. 0 = DISABLE 1 = ENABLE
1	0x0	CLR_CLK_ENB_CPULP: clear enable clock to CPULP. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CLK_ENB_CPUG: clear enable clock to CPUG. [CCLK Multi-Address] 0 = DISABLE 1 = ENABLE

6.5.146 CLK_RST_CONTROLLER_CLK_ENB_W_SET_0

Offset: 448h | Read/Write: R/W | Reset: 0b00000011111100

Bit	Reset	Description
13	0x0	SET_CLK_ENB_MIPI_IOBIST: set enable clock to MIPI IOBIST 0 = DISABLE 1 = ENABLE
12	0x0	SET_CLK_ENB_SATA_IOBIST: set enable clock to SATA IOBIST 0 = DISABLE 1 = ENABLE
11	0x0	SET_CLK_ENB_HDMI_IOBIST: set enable clock to HDMI IOBIST 0 = DISABLE 1 = ENABLE
10	0x0	SET_CLK_ENB_EMC_IOBIST: set enable clock to EMC IOBIST 0 = DISABLE 1 = ENABLE
9	0x0	SET_CLK_ENB_PCIE2_IOBIST: set enable clock to PCIE2 IOBIST 0 = DISABLE 1 = ENABLE
8	0x0	SET_CLK_ENB_CEC: set enable clock to CEC 0 = DISABLE 1 = ENABLE
7	0x1	SET_CLK_ENB_PCIERX5: set enable clock to PCIERX5 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x1	SET_CLK_ENB_PCIERX4: set enable clock to PCIERX4 0 = DISABLE 1 = ENABLE
5	0x1	SET_CLK_ENB_PCIERX3: set enable clock to PCIERX3 0 = DISABLE 1 = ENABLE
4	0x1	SET_CLK_ENB_PCIERX2: set enable clock to PCIERX2 0 = DISABLE 1 = ENABLE
3	0x1	SET_CLK_ENB_PCIERX1: set enable clock to PCIERX1 0 = DISABLE 1 = ENABLE
2	0x1	SET_CLK_ENB_PCIERX0: set 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 enable clock to HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

6.5.147 CLK_RST_CONTROLLER_CLK_ENB_W_CLR_0

Offset: 44ch | Read/Write: R/W | Reset: 0b00000011111100

Bit	Reset	Description
13	0x0	CLR_CLK_ENB_MIPI_IOBIST: clear enable clock to MIPI IOBIST 0 = DISABLE 1 = ENABLE
12	0x0	CLR_CLK_ENB_SATA_IOBIST: clear enable clock to SATA IOBIST 0 = DISABLE 1 = ENABLE
11	0x0	CLR_CLK_ENB_HDMI_IOBIST: clear enable clock to HDMI IOBIST 0 = DISABLE 1 = ENABLE
10	0x0	CLR_CLK_ENB_EMC_IOBIST: clear enable clock to EMC IOBIST 0 = DISABLE 1 = ENABLE
9	0x0	CLR_CLK_ENB_PCIE2_IOBIST: clear enable clock to PCIE2 IOBIST 0 = DISABLE 1 = ENABLE
8	0x0	CLR_CLK_ENB_CEC: clear enable clock to CEC 0 = DISABLE 1 = ENABLE
7	0x1	CLR_CLK_ENB_PCIERX5: clear enable clock to PCIERX5 0 = DISABLE 1 = ENABLE
6	0x1	CLR_CLK_ENB_PCIERX4: clear enable clock to PCIERX4 0 = DISABLE 1 = ENABLE
5	0x1	CLR_CLK_ENB_PCIERX3: clear enable clock to PCIERX3 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x1	CLR_CLK_ENB_PCIERX2: clear enable clock to PCIERX2 0 = DISABLE 1 = ENABLE
3	0x1	CLR_CLK_ENB_PCIERX1: clear enable clock to PCIERX1 0 = DISABLE 1 = ENABLE
2	0x1	CLR_CLK_ENB_PCIERX0: clear enable clock to PCIERX0 0 = DISABLE 1 = ENABLE
1	0x0	CLR_CLK_ENB_RESERVED0: reserved SATACOLD 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CLK_ENB_HDA2HDMICODEC: clear enable clock to HDA2HDMICODEC 0 = DISABLE 1 = ENABLE

6.5.148 CLK_RST_CONTROLLER_RST_CPUG_CMPLX_SET_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 450h | Read/Write: R/W | Reset: 0b000xxxxxxxxxxxx1110111011101110

Bit	Reset	Description
30	0x0	SET_PRESETDBG: 1 = assert nPRESETDBG to the coresight. 0 = DISABLE 1 = ENABLE
29	0x0	SET_SCURESET: 1 = assert nSCURESET to the SCU. 0 = DISABLE 1 = ENABLE
28	0x0	SET_PERIPHRESET: 1 = assert nPERIPHRESET to the CPU. 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	0x0	SET_DBGRESET0: 1 = assert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
11	0x1	SET_WDRESET3: 1 = assert nWDRESET to CPU3. 0 = DISABLE 1 = ENABLE
10	0x1	SET_WDRESET2: 1 = assert nWDRESET to CPU2. 0 = DISABLE 1 = ENABLE
9	0x1	SET_WDRESET1: 1 = assert nWDRESET to CPU1. 0 = DISABLE 1 = ENABLE
8	0x0	SET_WDRESET0: 1 = assert nWDRESET to CPU0. 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
7	0x1	SET_DERESET3: 1 = assert nDERESET to CPU3. 0 = DISABLE 1 = ENABLE
6	0x1	SET_DERESET2: 1 = assert nDERESET to CPU2. 0 = DISABLE 1 = ENABLE
5	0x1	SET_DERESET1: 1 = assert nDERESET to CPU1. 0 = DISABLE 1 = ENABLE
4	0x0	SET_DERESET0: 1 = assert nDERESET to CPU0. 0 = DISABLE 1 = ENABLE
3	0x1	SET_CPURESET3: 1 = assert nCPURESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	SET_CPURESET2: 1 = assert nCPURESET to CPU2. 0 = DISABLE 1 = ENABLE
1	0x1	SET_CPURESET1: 1 = assert nCPURESET to CPU1. 0 = DISABLE 1 = ENABLE
0	0x0	SET_CPURESET0: 1 = assert nCPURESET to CPU0. 0 = DISABLE 1 = ENABLE

6.5.149 CLK_RST_CONTROLLER_RST_CPUG_CMPLX_CLR_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 454h | Read/Write: R/W | Reset: 0b000xxxxxxxxxxx1110111011101110

Bit	Reset	Description
30	0x0	CLR_PRESETDBG: 1 = deassert nPRESETDBG to the coresight. 0 = DISABLE 1 = ENABLE
29	0x0	CLR_SCURESET: 1 = deassert nSCURESET to the SCU. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_PERIPHRESET: 1 = deassert nPERIPHRESET to the CPU's interrupt/timer. 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
13	0x1	CLR_DBGRESET1: 1 = deassert nDBGRESET to CPU1. 0 = DISABLE 1 = ENABLE
12	0x0	CLR_DBGRESET0: 1 = deassert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
11	0x1	CLR_WDRESET3: 1 = deassert nWDRESET to CPU3. 0 = DISABLE 1 = ENABLE
10	0x1	CLR_WDRESET2: 1 = deassert nWDRESET to CPU2. 0 = DISABLE 1 = ENABLE
9	0x1	CLR_WDRESET1: 1 = deassert nWDRESET to CPU1. 0 = DISABLE 1 = ENABLE
8	0x0	CLR_WDRESET0: 1 = deassert nWDRESET to CPU0. 0 = DISABLE 1 = ENABLE
7	0x1	CLR_DERESET3: 1 = deassert nDERESET to CPU3. 0 = DISABLE 1 = ENABLE
6	0x1	CLR_DERESET2: 1 = deassert nDERESET to CPU2. 0 = DISABLE 1 = ENABLE
5	0x1	CLR_DERESET1: 1 = deassert nDERESET to CPU1. 0 = DISABLE 1 = ENABLE
4	0x0	CLR_DERESET0: 1 = deassert nDERESET to CPU0. 0 = DISABLE 1 = ENABLE
3	0x1	CLR_CPURESET3: 1 = deassert nCPURESET to CPU3. 0 = DISABLE 1 = ENABLE
2	0x1	CLR_CPURESET2: 1 = deassert nCPURESET to CPU2. 0 = DISABLE 1 = ENABLE
1	0x1	CLR_CPURESET1: 1 = deassert nCPURESET to CPU1. 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CPURESET0: 1 = deassert nCPURESET to CPU0. 0 = DISABLE 1 = ENABLE

6.5.150 CLK_RST_CONTROLLER_RST_CPULP_CMPLX_SET_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 458h | Read/Write: R/W | Reset: 0b000xxxxxxxxxxx1110111011101110

Bit	Reset	Description
30	0x0	SET_PRESETDBG: 1 = assert nPRESETDBG to the coresight. 0 = DISABLE 1 = ENABLE
29	0x0	SET_SCURESET: 1 = assert nSCURESET to the SCU. 0 = DISABLE 1 = ENABLE
28	0x0	SET_PERIPHRESET: 1 = assert nPERIPHRESET to the CPU. 0 = DISABLE 1 = ENABLE
15	0x1	SET_DBGRESET3: 1 = assert nDBGRESET to CPU3. N/A in CPULP. Reserve and init to reset.

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
14	0x1	SET_DBGRESET2: 1 = assert nDBGRESET to CPU2. N/A in CPULP. Reserve and init to reset. 0 = DISABLE 1 = ENABLE
13	0x1	SET_DBGRESET1: 1 = assert nDBGRESET to CPU1. N/A in CPULP. Reserve and init to reset. 0 = DISABLE 1 = ENABLE
12	0x0	SET_DBGRESET0: 1 = assert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
11	0x1	SET_WDRESET3: 1 = assert nWDRESET to CPU3. N/A in CPULP. Reserve and init to reset. 0 = DISABLE 1 = ENABLE
10	0x1	SET_WDRESET2: 1 = assert nWDRESET to CPU2. N/A in CPULP. Reserve and init to reset. 0 = DISABLE 1 = ENABLE
9	0x1	SET_WDRESET1: 1 = assert nWDRESET to CPU1. N/A in CPULP. Reserve and init to reset. 0 = DISABLE 1 = ENABLE
8	0x0	SET_WDRESET0: 1 = assert nWDRESET to CPU0. 0 = DISABLE 1 = ENABLE
7	0x1	SET_DERESET3: 1 = assert nDERESET to CPU3. N/A in CPULP. Reserve and init to reset. 0 = DISABLE 1 = ENABLE
6	0x1	SET_DERESET2: 1 = assert nDERESET to CPU2. N/A in CPULP. Reserve and init to reset. 0 = DISABLE 1 = ENABLE
5	0x1	SET_DERESET1: 1 = assert nDERESET to CPU1. N/A in CPULP. Reserve and init to reset. 0 = DISABLE 1 = ENABLE
4	0x0	SET_DERESET0: 1 = assert nDERESET to CPU0. 0 = DISABLE 1 = ENABLE
3	0x1	SET_CPURESET3: 1 = assert nCPURESET to CPU3. N/A in CPULP. Reserve and init to reset. 0 = DISABLE 1 = ENABLE
2	0x1	SET_CPURESET2: 1 = assert nCPURESET to CPU2. N/A in CPULP. Reserve and init to reset. 0 = DISABLE 1 = ENABLE
1	0x1	SET_CPURESET1: 1 = assert nCPURESET to CPU1. N/A in CPULP. Reserve and init to reset. 0 = DISABLE 1 = ENABLE
0	0x0	SET_CPURESET0: 1 = assert nCPURESET to CPU0.

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE

6.5.151 CLK_RST_CONTROLLER_RST_CPULP_CMPLX_CLR_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 45ch | Read/Write: R/W | Reset: 0b000xxxxxxxxxxxxxxx0xxx0xxx0xxx0

Bit	Reset	Description
30	0x0	CLR_PRESETDBG: 1 = deassert nPRESETDBG to the coresight. 0 = DISABLE 1 = ENABLE
29	0x0	CLR_SCURESET: 1 = deassert nSCURESET to the SCU. 0 = DISABLE 1 = ENABLE
28	0x0	CLR_PERIPHRESET: 1 = deassert nPERIPHRESET to the CPU's interrupt/timer. 0 = DISABLE 1 = ENABLE
12	0x0	CLR_DBGRESET0: 1 = deassert nDBGRESET to CPU0. 0 = DISABLE 1 = ENABLE
8	0x0	CLR_WDRESET0: 1 = deassert nWDRESET to CPU0. 0 = DISABLE 1 = ENABLE
4	0x0	CLR_DERESET0: 1 = deassert nDERESET to CPU0. 0 = DISABLE 1 = ENABLE
0	0x0	CLR_CPURESET0: 1 = deassert nCPURESET to CPU0. 0 = DISABLE 1 = ENABLE

6.5.152 CLK_RST_CONTROLLER_CLK_CPUG_CMPLX_SET_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 460h | Read/Write: R/W | Reset: 0b0000

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

6.5.153 CLK_RST_CONTROLLER_CLK_CPUG_CMPLX_CLR_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 464h | Read/Write: R/W | Reset: 0b0000

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

6.5.154 CLK_RST_CONTROLLER_CLK_CPULP_CMPLX_SET_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 468h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
8	0x0	SET_CPU0_CLK_STP: 1 = assert CPU0 clock stop 0 = DISABLE 1 = ENABLE

6.5.155 CLK_RST_CONTROLLER_CLK_CPULP_CMPLX_CLR_0

[CCLK Multi-Address]: Refer to details in file header

Offset: 46ch | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
8	0x0	CLR_CPU0_CLK_STP: 1 = deassert CPU0 clock stop 0 = DISABLE 1 = ENABLE

6.5.156 CLK_RST_CONTROLLER_CPU_CMPLX_STATUS_0

Offset: 470h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	PRESETDBG: ENABLE = nPRESETDBG to the coresight asserted 0 = DISABLE 1 = ENABLE
29	X	SCURESET: ENABLE = nSCURESET to the SCU asserted 0 = DISABLE 1 = ENABLE
28	X	PERIPHRESET: ENABLE = nPERIPHRESET to the CPU asserted 0 = DISABLE 1 = ENABLE
27	X	CSITEPTMRESET: ENABLE = nCSITEPTMRESET asserted

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
26	X	AXICIFRESET: ENABLE = nAXICIFRESET asserted 0 = DISABLE 1 = ENABLE
25	X	MPCORERESET: ENABLE = nMPCORERESET to the CPU asserted 0 = DISABLE 1 = ENABLE
24	X	CPU_CMPLX_CLKEN: ENABLE = CPU_CMPLX_CLKEN to the CPU asserted 0 = DISABLE 1 = ENABLE
23	X	RST_CPU0_FROM_FLOW: ENABLE = RST_CPU0_FROM_FLOW to the CPU asserted 0 = DISABLE 1 = ENABLE
15	X	DBGRESET3: ENABLE = nDBGRESET to CPU3 asserted 0 = DISABLE 1 = ENABLE
14	X	DBGRESET2: ENABLE = nDBGRESET to CPU2 asserted 0 = DISABLE 1 = ENABLE
13	X	DBGRESET1: ENABLE = nDBGRESET to CPU1 asserted 0 = DISABLE 1 = ENABLE
12	X	DBGRESET0: ENABLE = nDBGRESET to CPU0 asserted 0 = DISABLE 1 = ENABLE
11	X	WDRESET3: ENABLE = nWDRESET to CPU3 asserted 0 = DISABLE 1 = ENABLE
10	X	WDRESET2: ENABLE = nWDRESET to CPU2 asserted 0 = DISABLE 1 = ENABLE
9	X	WDRESET1: ENABLE = nWDRESET to CPU1 asserted 0 = DISABLE 1 = ENABLE
8	X	WDRESET0: ENABLE = nWDRESET to CPU0 asserted 0 = DISABLE 1 = ENABLE
3	X	CPURESET3: ENABLE = nCPURESET to CPU3 asserted 0 = DISABLE 1 = ENABLE
2	X	CPURESET2: ENABLE = nCPURESET to CPU2 asserted 0 = DISABLE 1 = ENABLE
1	X	CPURESET1: ENABLE = nCPURESET to CPU1 asserted 0 = DISABLE 1 = ENABLE
0	X	CPURESET0: ENABLE = nCPURESET to CPU0 asserted 0 = DISABLE 1 = ENABLE

6.5.157 CLK_RST_CONTROLLER_INTSTATUS_0

Interrupt Status Register.

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

Offset: 478h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
19:0	0x0	CAR_INT: Clear on 1-write. Init value is clear.

6.5.158 CLK_RST_CONTROLLER_INTMASK_0

Interrupt Mask Register.

Offset: 47ch | Read/Write: R/W | Reset: 0b00000000000000000000

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

6.5.159 CLK_RST_CONTROLLER_UTMIP_PLL_CFG0_0

The data sampling frequency relies on a 960MHz clock, so the goal of the PLL is to have: In_Frequency*(PLL_VCOMULTBY2+1)*(PLL_NDIV/PLL_MDIV) = 960MHz With a 12MHz 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: 480h | Read/Write: R/W | Reset: 0b00000000010100000000000110000000

Bit	Reset	Description
30:28	0x0	UTMIP_PLL_SELECT: Selects which of the eight 60MHz clock phases to produce at the output of the USB PHY PLL. This is for a test mode. See the cell specification.
27	0x0	UTMIP_PLL_PDIVRST: PDIVRST of the UTMIP PHY PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.
26:24	0x0	UTMIP_PLL_PDIV: PDIV[2:0] input of the UTMIP PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See the cell specification.
23:16	0x28	UTMIP_PLL_NDIV: NDIV[7:0] input of UTMIP PLL. This is the feedback divider on the VCO feedback. 0x0 is not allowed. See the 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 the cell specification.
7	0x1	UTMIP_PLL_VCOMULTBY2: VCOMULTBY2 control of the UTMIP PHY PLL. Recommended setting is on. Additional divide by 2 on the VCO feedback. Which is setting the bit to 1. See the cell specification.
6:1	0x0	UTMIP_PLL_LOCKSEL: LOCKSEL[5:0] input of UTMIP PLL. Used in test modes only. See the cell specification.
0	0x0	UTMIP_PLL_LOCKENABLE: LOCK_ENABLE input of UTMIP PLL. Normally only used during test. See the cell specification.

6.5.160 CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0

PLL CONFIGURATION & PARAMETERS

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

Xtal clock -> enters PLL_U to generate 12MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock

The following parameters control the bringup of the PLLs:

Coming out of reset or suspend

-> PllUOnState: start pllu_enable_count and pll_lock_count

-> Wait ~1us to actually enable the PLL_U (pllu_enable_count == ClkXtal * PLLU_ENABLE_DLY_COUNT * 8)

-> Wait ~1ms until PLL_U is actually stable (pll_lock_count == ClkXtal * PLLU_STABLE_COUNT * 256) => USB_PHY PLL_ENABLE pll_active_count = 0

-> Wait 5us to enable pll_active: pll_active_count == ClkXtal * PLL_ACTIVE_DLY_COUNT * 16 => USB_PHY PLL_ACTIVE

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

Numbers for a 19.2MHz Xtal 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
- PLL_ACTIVE_DLY_COUNT[4:0] = 10 * 19.2 / 16 = 12 = 0x0c
- XTAL_FREQ_COUNT[11:0] = 2500 * 19.2 / 256 = 187 = 0xbb

UTMIP PLL and PLLU configuration register 1

Offset: 484h | Read/Write: R/W | Reset: 0b00011000001000010101000011000000

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	0x8	UTMIP_PLL_SETUP: SETUP[8:0] input of USB_PHY PLL.
17	0x0	UTMIP_FORCE_PLLU_POWERUP: Force PLL_U into power up.
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.

6.5.161 CLK_RST_CONTROLLER_UTMIP_PLL_CFG2_0

UTMIP Miscellaneous Configurations

Offset: 488h | Read/Write: R/W | Reset: 0b10101xx00110001100000000010101

Bit	Reset	Description
30	0x1	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
29	0x0	UTMIP_FORCE_PD_CLK60_POWERUP: Force UTMIP PLL PD_CLK60 input into power up.
28	0x1	UTMIP_FORCE_PD_CLK60_POWERDOWN: Force UTMIP PLL PD_CLK60 input into power down. (Overrides FORCE_PD_CLK60_POWERUP.)
27	0x0	UTMIP_FORCE_PD_CLK48_POWERUP: Force UTMIP PLL PD_CLK48 input into power up.
26	0x1	UTMIP_FORCE_PD_CLK48_POWERDOWN: Force UTMIP PLL PD_CLK48 input into power down. (Overrides FORCE_PD_CLK48_POWERUP.)
23:18	0xc	UTMIP_PLL_ACTIVE_DLY_COUNT: 10 us / (1/19.2MHz) = 192 / 16 = 12
17:6	0x600	UTMIP_PLLU_STABLE_COUNT: Reserved.
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 into power up.
2	0x1	UTMIP_FORCE_PD_SAMP_B_POWERDOWN: Force UTMIP PLL PD_SAMP_B input 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.

Bit	Reset	Description
0	0x1	UTMIP_FORCE_PD_SAMP_A_POWERDOWN: Force UTMIP PLL PD_SAMP_A input into power down. (Overrides FORCE_PD_SAMP_A_POWERUP.)

6.5.162 CLK_RST_CONTROLLER_PLLE_AUX_0

Note that Range value is the recommended range. All counters supports 0-255 step range.

Offset: 48ch | Read/Write: R/W | Reset: 0b0xxxxx10000001000100000001110000

Bit	R/W	Reset	Description
31	RW	SKIP	PLLE_SS_SEQ_INCLUDE: 0=skip, 1=include. PLLE spread spectrum power sequencer include. 0 = SKIP 1 = INCLUDE
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 0-255ms in 1ms steps.
15:8	RW	0x40	PLLE_LOCK_DLY: PT5: Delay from PLLE ENABLE to lock. Range 0-128us in 1us 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 us or ms, fast is per osc clock. Applies to all programmable timer counters in SATA, PCIE and PLLE seq.
6	RW	0x1	PLLE_SS_SWCTL: 0=PLLE spread spectrum config setup by HW, 1=setup by SW during training
5	RW	0x1	PLLE_CONFIG_SWCTL: 0=PLLE config setup by HW, 1=PLLE setup by SW during training. Affects resetable bits when PLL is off. SETUP and EXT_SETUP bits. All other config bit are left untouched as initialized. ex. M/N/P, SS coeff.
4	RW	0x1	PLLE_ENABLE_SWCTL: 0=PLLE enable by HW, 1=PLLE enable by SW
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), 0=Reserved (use pre-divM=18 for 12MHz reference) 0 = OSC_DIV 0 = Reserved
1	RW	DISABLE	PLLE_CML1_OEN: PLLE CML1 clock out enable. Used for SATA. 1=enable 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
0	RW	DISABLE	PLLE_CML0_OEN: PLLE CML0 clock out enable. Used for PCIE. 1=enable. 0 = DISABLE 1 = ENABLE

6.5.163 CLK_RST_CONTROLLER_SATA_PLL_CFG0_0

SATA power sequencer input SW control program guide

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

SATA power sequencer is driven by 3 primary reset and powerdown input signals: reset, lane_pd, padpll_pd.

Normal power up sequence is to deassert in the order of padpll_pd->lane_pd->reset.

Normal power down sequence is to assert in the order of reset->lane_pd->padpll_pd.

It is acceptable to assert all three at the same time.

It is acceptable for reset or lane_pd to return to powerup state in the middle of a powerdown sequence.

Offset: 490h | Read/Write: R/W | Reset: 0bxx10xxxxxxxxxxxxxxx0000x011

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 power up, 1=SW control padpll PD by setting this and SATA_SEQ_IN_SWCTL bit
6	RW	0x0	SATA_SEQ_LANE_PD_INPUT_VALUE: 0=IOPHY is power up, 1=SW control IOPHY PD by setting this and SATA_SEQ_IN_SWCTL bit
5	RW	0x0	SATA_SEQ_RESET_INPUT_VALUE: 0=SATA reset deasserted, 1=SW control reset by setting this and SATA_SEQ_IN_SWCTL bit
4	RW	0x0	SATA_SEQ_IN_SWCTL: 0=seq input by HW, 1=seq input by SW
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 power up, 1=SW control reset by setting this and SATA_PADPLL_RESET_SWCTL bit
0	RW	0x1	SATA_PADPLL_RESET_SWCTL: 0=reset by HW, 1=reset by SW

6.5.164 CLK_RST_CONTROLLER_SATA_PLL_CFG1_0

Note that Range value is the recommended range. All counters supports 0-255us range in 1us steps

Offset: 494h | Read/Write: R/W | Reset: 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 0-200us.
23:16	0x20	SATA_PADPLL_IDDQ2LANE_SLUMBER_DLY: PT3: Delay from SATA PAD PLL out of IDDQ to lane placed out of IDDQ. Range 0-64us.
15:8	0x20	SATA_PADPLL_PU_POST_DLY: PT2: Delay from RESET to SATA PAD PLL lock. Range 0-64us.
7:0	0x8	SATA_LANE_IDDQ2_PADPLL_IDDQ_DLY: PT1: Delay from placing lane in IDDQ to PAD PLL in IDDQ. Range 0-16us.

6.5.165 CLK_RST_CONTROLLER_PCIE_PLL_CFG_0

PCIE power sequencer input SW control program guide

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

PCIe power sequencer is driven by 1 primary reset input signal: reset.

Reset is expect to be held static until powerup or down sequence is complete.

Offset: 498h | Read/Write: R/W | Reset: 0bxx10xxxxxxxxxxxxxxxxx001101

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=SW control reset by setting this and PCIE_SEQ_IN_SWCTL bit
4	RW	0x0	PCIE_SEQ_IN_SWCTL: 0=seq input by HW, 1=seq input by SW
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 enable by HW, 1=xclk enable by SW
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 HW, 1=reset by SW

6.5.166 CLK_RST_CONTROLLER_PROG_AUDIO_DLY_CLK_0

- Clock doubler with 1X/2X select for: I2S0..4, SPDIF.
- Clock doublers with 1x/2x select use by clock controller for each of I2S0..4, SPDIF 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 is used to provide programmable delay for the clock doublers. Individual clock doubler is provided for each audio sync clock: I2S0..4, SPDIF.

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: 49ch | Read/Write: R/W | Reset: 0b00000001110111011101110111011101

Bit	Reset	Description
29	0x0	SPDIF_1X_SEL: Select SPDIF 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 clk doubler
19:16	0x7	SYNC_CLK_I2S4_DELCLK_SEL: 16 Taps of selectable delay for SYNC_CLK clk doubler
15:12	0x7	SYNC_CLK_I2S3_DELCLK_SEL: 16 Taps of selectable delay for SYNC_CLK clk doubler

Bit	Reset	Description
11:8	0x7	SYNC_CLK_I2S2_DELCLK_SEL: 16 Taps of selectable delay for SYNC_CLK clk doubler
7:4	0x7	SYNC_CLK_I2S1_DELCLK_SEL: 16 Taps of selectable delay for SYNC_CLK clk doubler
3:0	0x7	SYNC_CLK_I2S0_DELCLK_SEL: 16 Taps of selectable delay for SYNC_CLK clk doubler

6.5.167 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S0_0

Audio sync clock source select

Offset: 4a0h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable AUDIO SYNC CLK. Also
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

6.5.168 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S1_0

Audio sync clock source select

Offset: 4a4h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable AUDIO SYNC CLK
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

6.5.169 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S2_0

Audio sync clock source select

Offset: 4a8h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable AUDIO SYNC CLK
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

6.5.170 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S3_0

Audio sync clock source select

Offset: 4ach | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable AUDIO SYNC CLK
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

6.5.171 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_I2S4_0

Audio sync clock source select

Offset: 4b0h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable AUDIO SYNC CLK

Bit	Reset	Description
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

6.5.172 CLK_RST_CONTROLLER_AUDIO_SYNC_CLK_SPDIF_0

Audio sync clock source select

Offset: 4b4h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	SYNC_CLK_DIS: 0 = Enable AUDIO SYNC CLK
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

6.5.173 CLK_RST_CONTROLLER_PLLD2_BASE_0

Offset: 4b8h | Read/Write: R/W | Reset: 0b000xx0xxx000xx0000000001xxx01100

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 clk, 1 = disable reference clk. 28:28 Reserved as BASE_OVERRIDE bit in PLLP 0 = REF_ENABLE 1 = REF_DISABLE
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. 25:25 Reserved as DSIB_CLK_SRC field in PLLD
22:20	RW	0x0	PLLD2_DIVP: 0 = post divider (2^n).

Bit	R/W	Reset	Description
17:8	RW	0x1	PLLD2_DIVN: PLL feedback divider.
4:0	RW	0xc	PLLD2_DIVM: PLL input divider.

6.5.174 CLK_RST_CONTROLLER_PLLD2_MISC_0

Offset: 4bch | Read/Write: R/W | Reset: 0b00000000000000000000000100000000

Bit	Reset	Description
31	0x0	PLLD2_FO_MODE: 1 = 5-125MHz, 0 = 40-1000MHz.
30	0x0	PLLD2_CLKENABLE: 0 = disable, 1 = normal operation.
29:27	0x0	PLLD2_PTS: Base PLLD test output select.
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.

7.0 TIMERS

The Timer Controller is a 29-bit compound controller. It counts ticks and evaluates/assigns a passage of time to the number of ticks.

Tegra® 3 devices have ten 29-bit timers that run at 1 MHz clock rate based on the crystal oscillator. Each timer can be programmed to generate single, periodic, or watchdog interrupts.

When the EN bit is set high, the timer loads the timer present trigger value (PTV) into its counter and starts decrementing at a 1 MHz clock rate (1 µsec increment). When the timer present count value (PCR) decrements to zero, it generates a timer interrupt. When the enable periodic interrupt (PER) bit is enabled, the timer generates an interrupt and reloads the counter with the PTV value and starts to decrement again. When the timer consecutively decrements to zero a second time, without the processor reading the first interrupt status register (PCR), it generates a watchdog timer (WDT) interrupt.

Watchdog timers facilitate recovery from system lockup conditions. Either of the two generic timers can be configured as the watchdog timer using the Reset Trigger Source (RST_SOURCE) register. The watchdog timer interrupts the processor when the selected counter expires. Under normal operation, this interrupt is serviced by the processor by reading the status register (timer register).

The Tegra 3 device adds five new watchdog timers, directly associated with the four CPU cores and the AVP. Any of the general-purpose timers can be used as a reference.

If the interrupt is not serviced by the processor before the watchdog timer counts down to zero a second time, this is treated as a lockup condition. When this occurs, the watchdog timer generates a reset to the system. Watchdog timer functionality is enabled using bit 5 (WDT.E) of the RST_SOURCE register. Depending on which timer is configured for the watchdog timer function, the processor needs to service the watchdog interrupts by reading the corresponding timer status registers TMR1_PCR and TMR2_PCR. The reset generated by the watchdog timer can be configured to reset the system (entire chip) or the processors individually. This configuration is done using bits [2:0] of the RST_SOURCE register. Software can determine the cause of the reset by checking the status flags (bits [13:8]) of the RST_SOURCE. The USEC_DIVIDEND" and "USEC_DIVISOR" are used to indicate what fraction of 1 microsecond each oscillator clock represents.

Note:	If the XTAL oscillator or clock source is disabled during low-power standby mode, the microsecond timers will not function. This also means that interrupts will not occur.

7.1 Functionality

The programmable timer block contains ten 29-bit programmable timer counters and a 32-bit time-stamp counter.

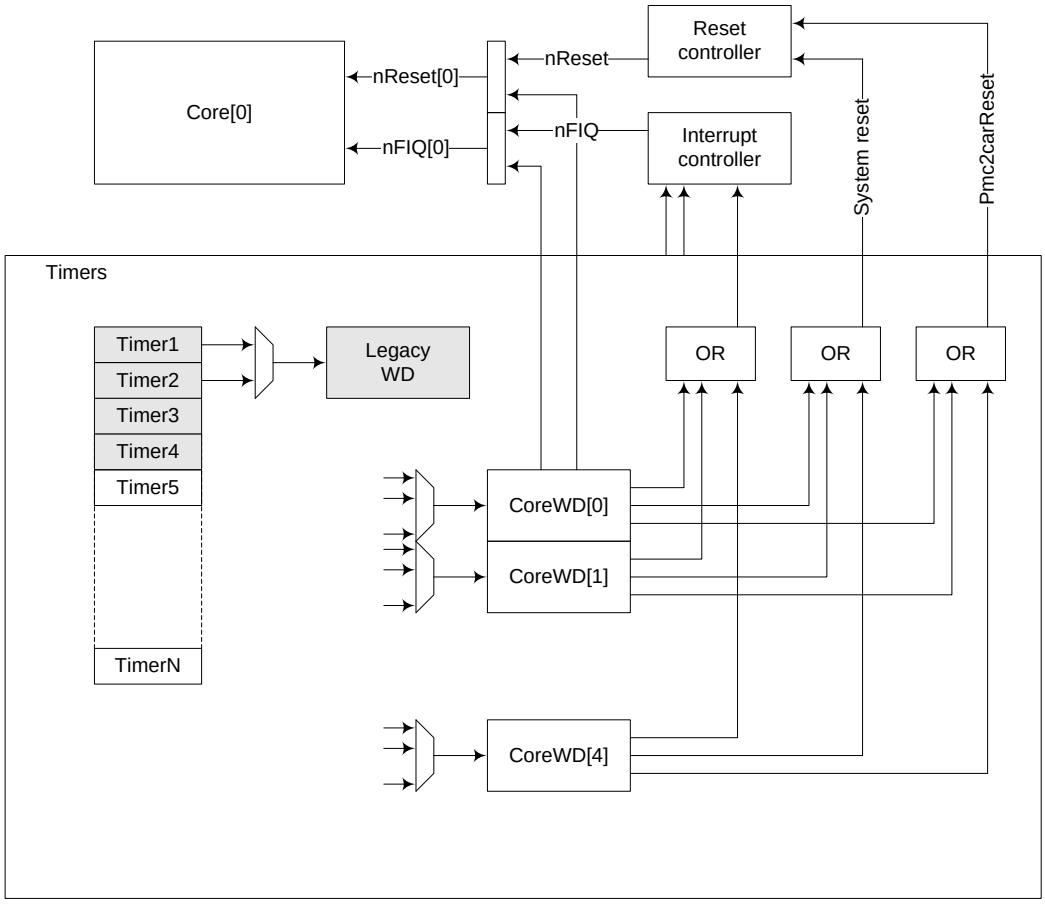
Each Timer can be programmed to generate a periodic interrupt or a one-time interrupt. The Timer count decrements by one microsecond when a timer is programmed via a corresponding Enable-bit. The timer generates a timer request whenever the count reaches Zero. If the periodic control bit is set, the timer will reload itself with the count value and decrement again. If the Periodic bit is not set, the timer will generate a single interrupt, clear its enable bit and not reload.

The timer count is 29 bits wide, therefore, it will support a periodic interrupt every 16.384 ms maximum or a minimum of 1 µs. 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 Decrement counter. The time-stamp 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. This reference decomposition is for documentation purposes only and the designer may choose a different structure for HW implementation as long as functional requirements are met.

Figure 5. Timer Functional Block Diagram



- The timers are numbered 0 to 9
- Timers 1 to 5 have a dedicated interrupt bit
- Timers 0,6,7,8,9 share a common interrupt called SharedTimerInterrupt.

7.2 Watchdog Timer Programming Guide

The watchdog timer function facilitates recovery from system lockup. The watchdog 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 will run the normal boot code. It would be up to the boot loader to decide if it needs to restart the CPU as well.

The watchdog timer can be programmed to a maximum timeout value of 0x3fffffff which is more than 1 billion. At 1µs/count, this is more than one thousand seconds.

It is operated as follows:

1. Watchdog generates Interrupt based on 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. Processor must handle interrupt or Reset is issued

a. First timeout -- interrupt is generated

b. If interrupt is not handled by next timeout, reset issued

3. Reset can be for AVP, CPU, or full system

a. CAR.RST_SOURCE.WDT_***_RST_EN = 1

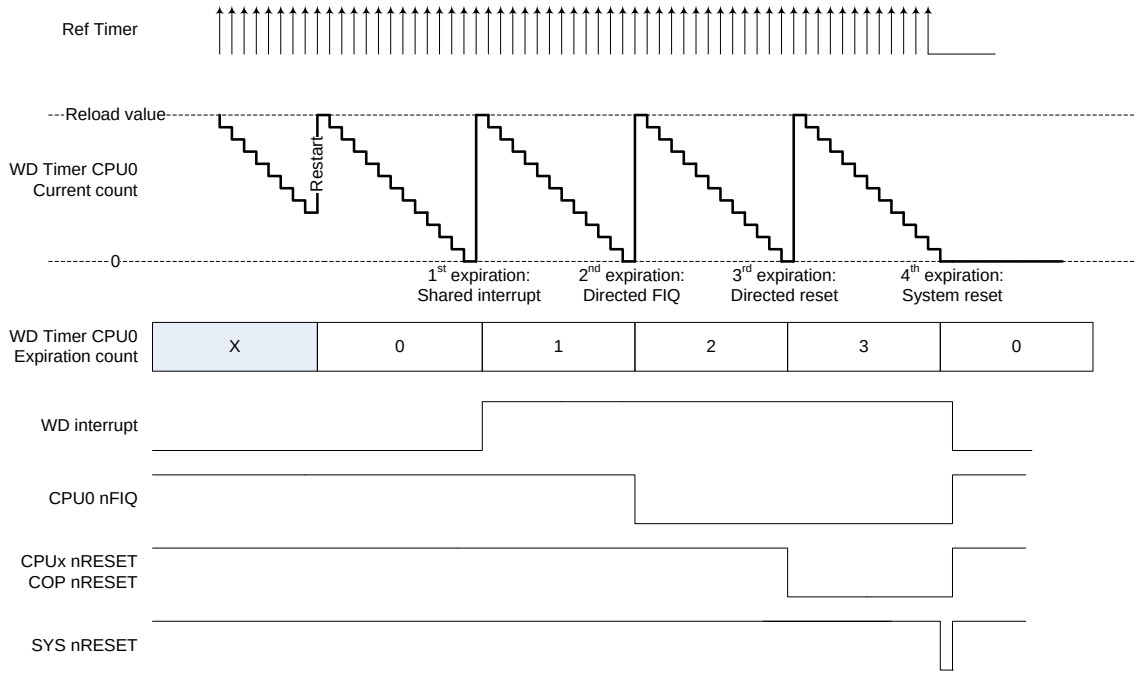
4. Upon reboot, CAR.RST_SOURCE.WDT_*_RST_STA indicates cause of the reset

Tegra 3 devices have 5 watchdog timers that operate in the following way:

- Directly associated with a processor core, i.e., four 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
- 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 HW reset)

[1] CPU Core 0 is treated in a specific manner so that the watchdog process is transparent to SW migration from the G to the LP process.

Figure 6. Watchdog Example Timing Diagram

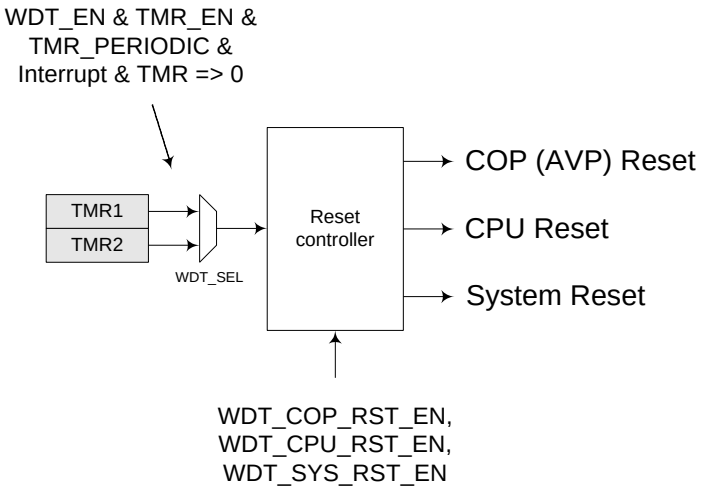


Any operation that targets CPU0 is done in the following way:

- 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 7. Watchdog Block Diagram



7.3 Timer Registers

7.3.1 TIMER_TMR1_TMR_PTV_0

Before programming timer:

- Input oscillator frequency must be specified in the TIMERUS_USEC_CFG to support multiple frequencies
- Program the number of 1us 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, 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: 000h | Read/Write: R/W | Reset: 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.

7.3.2 TIMER_TMR1_TMR_PCR_0

When interrupt is generated, write "1" to PCR[30] present count Register to clear the interrupt

Timer interrupt assignments:

- 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

Timer Present Count Value (Status) Register

Offset: 004h | Read/Write: R/W | Reset: 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.

7.3.3 TIMER_TMR2_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 008h | Read/Write: R/W | Reset: 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.

7.3.4 TIMER_TMR2_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 00ch | Read/Write: R/W | Reset: 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.

7.3.5 TIMER_TMR3_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 050h | Read/Write: R/W | Reset: 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.

7.3.6 TIMER_TMR3_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 054h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
30	RW	0x0	INTR_CLR: 1 = clears the interrupt, 0 = no affect. Write-1-to-Clear

Bit	R/W	Reset	Description
28:0	RO	X	TMR_PCV: Counter value: decrements from PTV.

7.3.7 TIMER_TMR4_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 058h | Read/Write: R/W | Reset: 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.

7.3.8 TIMER_TMR4_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 05ch | Read/Write: R/W | Reset: 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.

7.3.9 TIMER_TMR5_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 060h | Read/Write: R/W | Reset: 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.

7.3.10 TIMER_TMR5_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 064h | Read/Write: R/W | Reset: 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.

7.3.11 TIMER_TMR6_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 068h | Read/Write: R/W | Reset: 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.

7.3.12 TIMER_TMR6_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 06ch | Read/Write: R/W | Reset: 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.

7.3.13 TIMER_TMR7_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 070h | Read/Write: R/W | Reset: 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.

7.3.14 TIMER_TMR7_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 074h | Read/Write: R/W | Reset: 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.

7.3.15 TIMER_TMR8_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 078h | Read/Write: R/W | Reset: 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.

7.3.16 TIMER_TMR8_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 07ch | Read/Write: R/W | Reset: 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.

7.3.17 TIMER_TMR9_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 080h | Read/Write: R/W | Reset: 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.

7.3.18 TIMER_TMR9_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 084h | Read/Write: R/W | Reset: 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.

7.3.19 TIMER_TMR0_TMR_PTV_0

Timer Present Trigger Value (Set) Register

Offset: 088h | Read/Write: R/W | Reset: 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.

7.3.20 TIMER_TMR0_TMR_PCR_0

Timer Present Count Value (Status) Register

Offset: 08ch | Read/Write: R/W | Reset: 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.

7.4 Timer USEC CFG

The purpose of 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).

7.4.1 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 micro-second.

"USEC_DIVIDEND" and "USEC_DIVISOR" are used to indicate what fraction of 1 microsecond each oscillator clock represents.

Table 22 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

7.5 CNTR_1US Register

This free-running read-only register/counter changes once every micro-second and is used mainly by hardware (can also be used by software). It starts counting from 0 once it's 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 micro-second before a flow control condition is triggered.
- Used by secure boot logic.

7.5.1 TIMERUS_CNTR_1US_0

Offset: 000h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	x	HIGH_VALUE: Elapsed time in micro-second
15:0	x	LOW_VALUE: Elapsed time in micro-second

7.5.2 TIMERUS_USEC_CFG_0

Offset: 004h | Read/Write: R/W | Reset: 0b0000000000001100

Bit	Reset	Description
15:8	0x0	USEC_DIVIDEND: usec dividend. (n+1)
7:0	0xc	USEC_DIVISOR: usec divisor. (n+1)

7.5.3 TIMERUS_CNTR_FREEZE_0

The timer disable logic freezes (stops incrementing or decrementing) the timer counters when one or both of the ARM cores enters debug state. The timers remain frozen while either core is in debug state or its freeze enable bit is asserted:

freeze = (cpu_timer_freeze && cpu_debug_state) || (cop_timer_freeze && cop_debug_state)

Offset: 03ch | Read/Write: R/W | Reset: 0b00000

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.

7.6 WatchDog Timers

- WDT0 dedicated to CPU0 (either cluster)
- WDT1 dedicated to CPU1
- WDT2 dedicated to CPU2
- WDT3 dedicated to CPU3
- WDT4 dedicated to COP (AVP)

Offset relative to TMR block base.

Each Watchdog timer has the following registers

- Timer sources are numbers 1 thru 10.
- 0 not used and results in undefined behavior
- Period of 0 results in MAX period
- The generated interrupt is dependent on 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

7.6.1 TIMER_WDT0_CONFIG_0

Core Watchdog Configuration Register

Offset: 100h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
20:16	0x0	CoreResetBitmapEn: Enable reset of a set of cores at 3rd 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 4th expiration of the counter, at 0 = DISABLE 1 = ENABLE
14	0x0	SystemResetEnable: Enable system reset at 4th expiration of the counter, at same level as existing watchdog interrupt 0 = DISABLE 1 = ENABLE
13	0x0	FIQEnable: Enable FIQ assertion at 2nd 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

7.6.2 TIMER_WDT0_STATUS_0

Core Watchdog Status

Offset: 104h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxx

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

7.6.3 TIMER_WDT0_COMMAND_0

When the StartCounter bit is set, it enables the counter operation, loads the counter with Period and starts downcounting, 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: 108h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	DisableCounter: Write 1 to disable WDT counter if unlocked
0	0x0	StartCounter: Write 1 to enable WDT counter

7.6.4 TIMER_WDT0_UNLOCK_PATTERN_0

CoreWatchdogDisableUnlock Register

When the StartCounter bit is set, it enables the counter operation, loads the couner with Period and starts downcounting, 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.

Offset: 10ch | Read/Write: R/W | Reset: 0b0000000000000000

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.

7.6.5 TIMER_WDT1_CONFIG_0

Offset: 120h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
20:16	0x0	CoreResetBitmapEn: Enable reset of a set of cores at 3rd 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 4th expiration of the counter, at 0 = DISABLE 1 = ENABLE
14	0x0	SystemResetEnable: Enable system reset at 4th expiration of the counter, at same level as existing watchdog interrupt 0 = DISABLE 1 = ENABLE
13	0x0	FIQEnable: Enable FIQ assertion at 2nd expiration of the counter 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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

7.6.6 TIMER_WDT1_STATUS_0

CoreWatchdogStatus

Offset: 124h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

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

7.6.7 TIMER_WDT1_COMMAND_0

CoreWatchdogCommand Register

Offset: 128h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	DisableCounter: Write 1 to disable WDT counter if unlocked
0	0x0	StartCounter: Write 1 to enable WDT counter

7.6.8 TIMER_WDT1_UNLOCK_PATTERN_0

CoreWatchdogDisableUnlock Register

Offset: 12ch | Read/Write: R/W | Reset: 0b0000000000000000

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.

7.6.9 TIMER_WDT2_CONFIG_0

Offset: 140h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
20:16	0x0	CoreResetBitmapEn: Enable reset of a set of cores at 3rd 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 4th expiration of the counter, at 0 = DISABLE 1 = ENABLE
14	0x0	SystemResetEnable: Enable system reset at 4th expiration of the counter, at same level as existing watchdog interrupt 0 = DISABLE 1 = ENABLE
13	0x0	FIQEnable: Enable FIQ assertion at 2nd 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

7.6.10 TIMER_WDT2_STATUS_0

CoreWatchdogStatus

Offset: 144h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

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

7.6.11 TIMER_WDT2_COMMAND_0

CoreWatchdogCommand Register

Offset: 148h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	DisableCounter: Write 1 to disable WDT counter if unlocked
0	0x0	StartCounter: Write 1 to enable WDT counter

7.6.12 TIMER_WDT2_UNLOCK_PATTERN_0

CoreWatchdogDisableUnlock Register

Offset: 14ch | Read/Write: R/W | Reset: 0b0000000000000000

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.

7.6.13 TIMER_WDT3_CONFIG_0

CoreWatchdogConfiguration Register

Offset: 160h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
20:16	0x0	CoreResetBitmapEn: Enable reset of a set of cores at 3rd 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 4th expiration of the counter, at 0 = DISABLE 1 = ENABLE
14	0x0	SystemResetEnable: Enable system reset at 4th expiration of the counter, at same level as existing watchdog interrupt 0 = DISABLE 1 = ENABLE
13	0x0	FIQEnable: Enable FIQ assertion at 2nd 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

7.6.14 TIMER_WDT3_STATUS_0

CoreWatchdogStatus

Offset: 164h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

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

7.6.15 TIMER_WDT3_COMMAND_0

CoreWatchdogCommand Register

Offset: 168h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	DisableCounter: Write 1 to disable WDT counter if unlocked
0	0x0	StartCounter: Write 1 to enable WDT counter

7.6.16 TIMER_WDT3_UNLOCK_PATTERN_0

CoreWatchdogDisableUnlock Register

Offset: 16ch | Read/Write: R/W | Reset: 0b0000000000000000

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.

7.6.17 TIMER_WDT4_CONFIG_0

Offset: 180h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
20:16	0x0	CoreResetBitmapEn: Enable reset of a set of cores at 3rd 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 4th expiration of the counter, at 0 = DISABLE 1 = ENABLE
14	0x0	SystemResetEnable: Enable system reset at 4th expiration of the counter, at same level as existing watchdog interrupt 0 = DISABLE 1 = ENABLE
13	0x0	FIQEnable: Enable FIQ assertion at 2nd 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

7.6.18 TIMER_WDT4_STATUS_0

CoreWatchdogStatus

Offset: 184h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

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

7.6.19 TIMER_WDT4_COMMAND_0

CoreWatchdogCommand Register

Offset: 188h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	DisableCounter: Write 1 to disable WDT counter if unlocked
0	0x0	StartCounter: Write 1 to enable WDT counter

7.6.20 TIMER_WDT4_UNLOCK_PATTERN_0

CoreWatchdogDisableUnlock Register

Offset: 18ch | Read/Write: R/W | Reset: 0b0000000000000000

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.

7.7 Timer Shared Interrupt Status

TimerSrcBitmap is used for Timers 6 thru 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.

7.7.1 SHARED_INTR_STATUS_0

Offset: 000h | Read/Write: RO | Reset: 0bxxxxxxxxxxx

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

8.0 MULTI-PURPOSE I/O PINS AND PIN MULTIPLEXING (PINMUXING)

Note: PRELIMINARY pending review

8.1 Overview

Many of the pins on Tegra® 3 devices are connected to multi-purpose I/O (MPIO) pads. Each MPIO can function as a GPIO. Additionally, each MPIO supports up to four Special-Function I/O (SFIO) roles. For example, the ball named LCD_CS0_N can act as the following:

- A GPIO
- The signal LCSN for the first LCD controller
- The signal LCSN for the second LCD controller
- The signal CS2 for SPI5A (i.e. interface A of SPI controller 5)

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 on Tegra 3 devices includes the “pin multiplexing” logic and registers to select a particular function for each MPIO.

This section describes the following features of MPIOs and the pinmux 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 and
- Recommendations for software programming related to the MPIO pads.

This section covers Tegra 3 technology Multi-Purpose digital I/O pads. It does not address the special-purpose I/O pads on Tegra 3 devices such as those used for USB, IC_USB, SATA, PCIE, TVO/DAC, MIPI DSI, MIPI CSI, the oscillator, or DRAM interfaces.

8.2 Terms and Acronyms

Some common terms used with Tegra 3 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

8.3 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
  - Either Schmitt mode, CMOS mode, or both
- A weak pull-up and a weak pull-down

Though each MPIO pad shares a similar structure, there are in fact several varieties of such pads. The varieties are designed to minimize the number of on-board components (such as level shifters or pull-up resistors) required in designs based on the Tegra 3 devices. The following table summarizes the differences between the five MPIO pad types.

Table 23: 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
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	4 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, down only. LPMD
ST	1.8, 2.8-3.3	Schmitt & CMOS	push-pull	VDDIO	50 kΩ or 100 kΩ	2 bits, up & down	5 bits, up & down. 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 to 3.3V (regardless of supply voltage) as long as the pad's output-driver is set to open-drain mode. During power-up sequencing, do NOT pull up a DD pad's IO until after both VDD_CORE and VDD_RTC are powered.

Note: Refer to Section 2.5, Power Sequencing, in *NVIDIA Tegra 3 Mobile Quad-Core Processors Data Sheet* (DS-05160) for a complete description of the power-up sequencing requirements for Tegra 3 processors.

The OD (open drain) MPIO pads are optimized to tolerate 5V on the IO 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 similar to the ST pads except for changes in the drive strength circuitry and in the weak pull-ups/pull-downs. Tegra 3 processors include CZ pads on the VDDIO_SDMMC1 and VDDIO_SDDMC3 power rails. Each of those rails also includes a pair of SDMEMCOMP pads. When the CZ pads are used with an SDMMC controller, a state machine within the SDMMC controller can adjust the output impedance of the CZ pads to match the resistors attached to the corresponding SDMEMCOMP pads.

Note: See SDMMC_SDMEMCOMPPADCTRL_0 in SDMMC section of this document, for more information.

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 3 processors include LV pads on the VDDIO_SDMMC4

and VDDIO_VI power rails. Each of those rails also includes LV_VTTGEN cells designed to protect the LVDIO pads from damage when the supply voltage is 1.8V.

Note: Refer to Section 4.2.1, Multi-Purpose I/O Pads, in *NVIDIA Tegra 3 Mobile Quad-Core Processors Data Sheet* (DS-05160) for the list of the pad type, the nominal pull-up/-down strength, and the IO power rail information associated with each MPIO. See the columns labeled “Pad Type”, “Pull Strength”, and “Power Rail.”

8.4 Pad Controls

The Tegra 3 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.

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
E_INPUT	Input Receiver (Enable/Disable): Enables or disables input receiver.
OD	Open Drain option: (applies to DD pads only) Selects between open-drain output driver and push-pull driver.
IO_RESET	Input/Output Reset: (applies to LV pads only). See section 8.8.5

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, PMC bypasses and then resets the pinmux controller registers. Software should reprogram these registers as necessary after returning from deep sleep.

The MPIO pads are partitioned into 41 “pad control groups”. The following controls can be configured independently for each pad control group.

HSM	High Speed Mode (Enable/Disable)
SCHMT	Schmitt Trigger (Enable/Disable): (Not applicable to LV pads)
LPMD	Low Power Mode
DRVDN / UP	Drive Down / Up
SLWR/ SLWF	Slew Falling / Rising

The controls are configured via the “pad control 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 24 lists the register address for each pad control group. Additionally, the table describes the bit positions of the controls within each register.

Table 26 lists the pad control group for each MPIO.

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 24. Pad Control Groups Register Addresses

Pad Control Group	Register Address	HSM	SCHMT	LPMD	DRVDN	DRVUP	SLWR	SLWF
Csuscfg	0x7000088c				16:12	23:19	27:24	31:28
Gmacfg	0x70000900				18:14	23:19	27:24	31:28
Gmbcfg	0x70000904				18:14	23:19	27:24	31:28
Gmccfg	0x70000908				18:14	23:19	27:24	31:28
Gmdcfg	0x7000090c				18:14	23:19	27:24	31:28
vicfg1	0x700008c8				18:14	23:19	27:24	31:28
sdio1cfg	0x700008ec	2	3		18:12	26:20	29:28	31:30
sdio2cfg	0x700008ac	2	3		18:12	26:20	29:28	31:30
sdio3cfg	0x700008b0	2	3		18:12	26:20	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
Crtcfg	0x700008f8	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
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
gpvcfg	0x70000928	2	3	5:4	16:12	24:20	29:28	31:30
lcdcfg1	0x700008a4	2	3	5:4	16:12	24:20	29:28	31:30
lcdcfg2	0x700008a8	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	18:14	23:19	25:24	29:28
atcfg2	0x70000874	2	3	5:4	18:14	23:19	25:24	29:28
atcfg3	0x70000878	2	3	5:4	18:14	23:19	29:28	31:30
atcfg4	0x7000087c	2	3	5:4	18:14	23:19	29:28	31:30
atcfg5	0x70000880	2	3	5:4	18:14	23:19	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

Note: The “dbgcfg” pad control group includes pads on both the VDDIO_SYS and VDDIO_UART power rails. Consider tying those two rails together if accurate drive strength is required.

Note: The “uabcfg” pad control group includes pads on the VDDIO_BB power rail and two pads on the VDDIO_SDMMC1 rail. Consider tying those two rails together (or not using the affected VDDIO_SDMMC1 pads) if accurate drive strength is required.

8.5 Pinmuxing

Tegra 3 processors include many types of IO controllers (such as I2C, SDMMC, NAND, and GMI). For some of these controller types, Tegra 3 processors include multiple “controller instances”. For example, Tegra 3 processors include five I2C controller instances.

Each controller instance on a Tegra 3 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 I2C 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.

Though each MPIO has up to five functions (such as a GPIO function and up to four SFIO functions), a given MPIO can only act as a single function at a given point in time. The pinmux controller on a Tegra 3 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, UART1’s TXD signal is available on the MPIO’s whose ball names are GPIO_PU1, UART2_RTS_N, ULPIO_DATA0, etc. In a system which brings out UART1’s TXD signal on the GPIO_PU1 ball, the pinmux registers for UART2_RTS_N and ULPI_DATA0 should be programmed to select some other signal.

Some controller instances make their entire set of signals available on two or more sets of MPIO’s. To say it another way, 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.** For example, the SDMMC4 controller has two interfaces – SDMMC4A and SDMMC4B. In a system which uses the SDMMC4A interface, all of the pinmux controller registers should be programmed to avoid selecting any signal from the SDMMC4B interface.

The pinmux controller includes one register per MPIO. Note that this is a significant change from Tegra 2 devices. In Tegra 2 devices, pinmuxing operated on a per-group basis. In Tegra 3 devices, it can be controlled on a per-pin basis.

Table 25 describes the layout of each pinmux controller register. Refer to the Excel spreadsheet, *Tegra 3 PinMux* (DA-05692) for a list of the pinmux register address and the default values for the pinmux register fields for each MPIO. See the columns labeled “pinmux register”, “PINMUX”, “TRISTATE”, “E_INPUT”, and “PUPD”.

Table 26 provides the register address for each pad that can be used for pinmuxing. It also lists the pad control group for each pad.

Figure 8 shows the pinmux logic associated with a single MPIO.

Figure 8. Structure of Pinmux Controller (per MPIO)

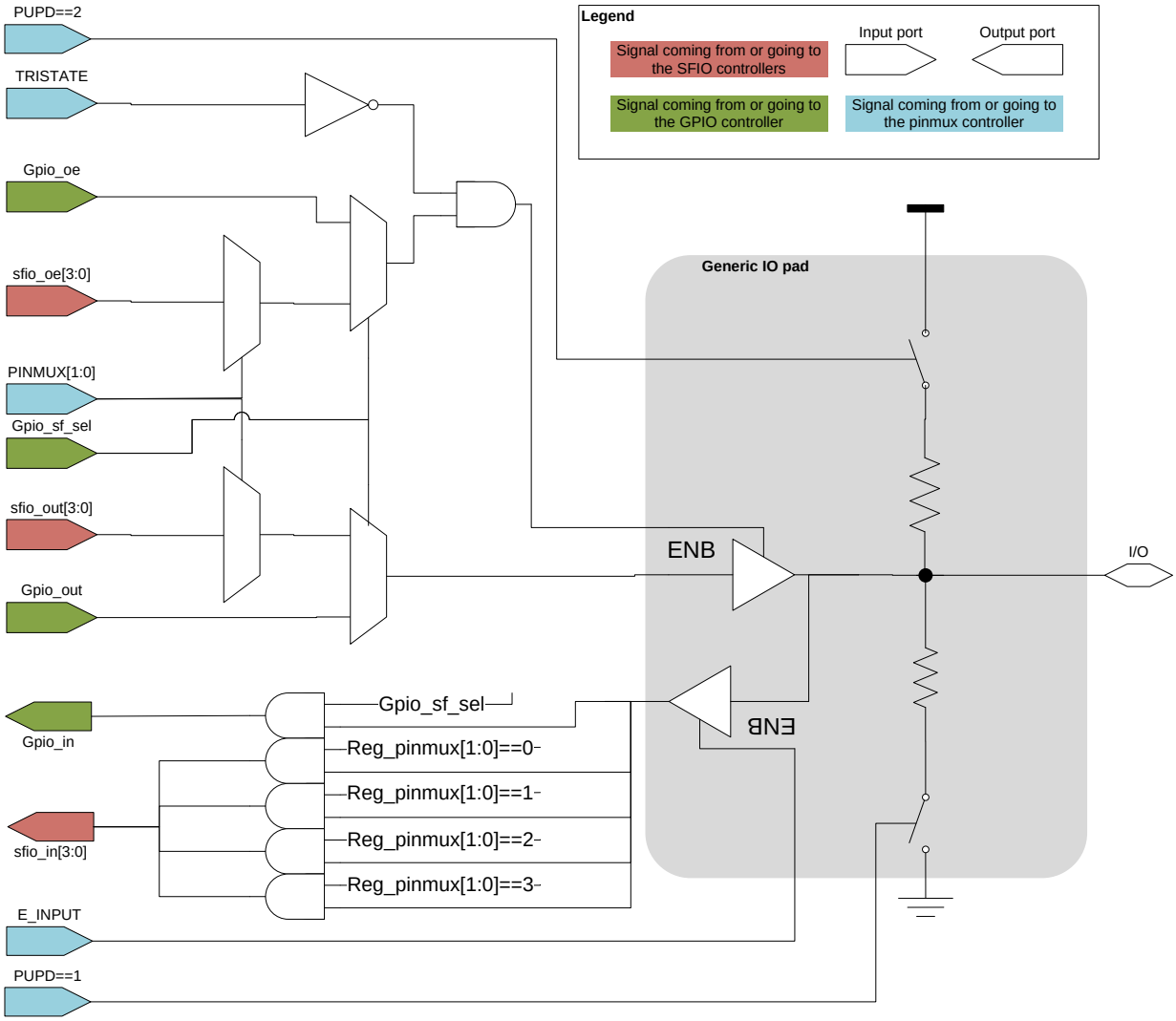


Table 25: Pinmux Register Format

Field Name	Bit Position	Default Value	Description
IO_RESET	8	LV pads: 1	Software should clear this bit after correctly programming the relevant E_18V bit in PMC. See the section in this chapter called “VDDIO_VI, VDDIO_SDMMC4, PMC, and IO_RESET” for more information.
		Other pads: 0	This bit only matters for LV pads. It is not implemented for other MPIO pad types.
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 & 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 IO power supply voltage.
		Other pads: 0	This bit only matters for DD pads. It is not implemented for other MPIO pad types.
E_INPUT	5	Refer to the Excel spreadsheet, *Tegra 3 PinMux* (DA-05692)	0: the pad’s input receiver is disabled. 1: the pad’s input receiver is enabled.
TRISTATE	4	Refer to the Excel spreadsheet, *Tegra 3 PinMux* (DA-05692)	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).
PUPD	3:2	Refer to the Excel spreadsheet, *Tegra 3 PinMux* (DA-05692)	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.
PINMUX	1:0	Refer to the Excel spreadsheet, *Tegra 3 PinMux* (DA-05692)	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.

Table 26. Pinmux Register Address

Ball name	register address	pad control group
CAM_I2C_SCL	0x70003290	gmecfg
CAM_I2C_SDA	0x70003294	gmecfg
CAM_MCLK	0x70003284	gmgcfg
CLK_32K_IN	0x70003330	aocfg2
CLK_32K_OUT	0x7000331c	aocfg2
CLK1_OUT	0x7000334c	cdev1cfg
CLK1_REQ	0x70003348	cdev1cfg
CLK2_OUT	0x70003068	cdev2cfg
CLK2_REQ	0x7000306c	cdev2cfg
CLK3_OUT	0x700031b8	dev3cfg
CLK3_REQ	0x700031bc	dev3cfg
CORE_PWR_REQ	0x70003324	aocfg2
CPU_PWR_REQ	0x70003328	aocfg2
CRT_HSYNC	0x7000311c	crtcfg
CRT_VSYNC	0x70003120	crtcfg
DAP1_DIN	0x7000333c	dap1cfg
DAP1_DOUT	0x70003340	dap1cfg
DAP1_FS	0x70003338	dap1cfg
DAP1_SCLK	0x70003344	dap1cfg
DAP2_DIN	0x7000335c	dap2cfg
DAP2_DOUT	0x70003360	dap2cfg
DAP2_FS	0x70003358	dap2cfg
DAP2_SCLK	0x70003364	dap2cfg
DAP3_DIN	0x70003034	dap3cfg
DAP3_DOUT	0x70003038	dap3cfg
DAP3_FS	0x70003030	dap3cfg
DAP3_SCLK	0x7000303c	dap3cfg
DAP4_DIN	0x700031ac	dap4cfg
DAP4_DOUT	0x700031b0	dap4cfg
DAP4_FS	0x700031a8	dap4cfg
DAP4_SCLK	0x700031b4	dap4cfg
DDC_SCL	0x70003114	ddccfg
DDC_SDA	0x70003118	ddccfg
GEN1_I2C_SCL	0x700031a4	dbgcfg
GEN1_I2C_SDA	0x700031a0	dbgcfg
GEN2_I2C_SCL	0x70003250	atcfg5
GEN2_I2C_SDA	0x70003254	atcfg5
GMI_A16	0x70003230	atcfg4
GMI_A17	0x70003234	atcfg4
GMI_A18	0x70003238	atcfg4
GMI_A19	0x7000323c	atcfg4
GMI_AD0	0x700031f0	atcfg2
GMI_AD1	0x700031f4	atcfg2
GMI_AD10	0x70003218	atcfg1
GMI_AD11	0x7000321c	atcfg1
GMI_AD12	0x70003220	atcfg1
GMI_AD13	0x70003224	atcfg1
GMI_AD14	0x70003228	atcfg1
GMI_AD15	0x7000322c	atcfg1
GMI_AD2	0x700031f8	atcfg2
GMI_AD3	0x700031fc	atcfg2
GMI_AD4	0x70003200	atcfg2
GMI_AD5	0x70003204	atcfg2
GMI_AD6	0x70003208	atcfg2
GMI_AD7	0x7000320c	atcfg2
GMI_AD8	0x70003210	atcfg1
GMI_AD9	0x70003214	atcfg1
GMI_ADV_N	0x700031cc	atcfg2
GMI_CLK	0x700031d0	atcfg2
GMI_CS0_N	0x700031d4	atcfg3
GMI_CS1_N	0x700031d8	atcfg4
GMI_CS2_N	0x700031dc	atcfg2
GMI_CS3_N	0x700031e0	atcfg2
GMI_CS4_N	0x700031e4	atcfg2
GMI_CS6_N	0x700031e8	atcfg2
GMI_CS7_N	0x700031ec	atcfg1
GMI_DQS	0x70003248	atcfg2
GMI_IORDY	0x700031c4	atcfg1
GMI_OE_N	0x70003244	atcfg2
GMI_RST_N	0x7000324c	atcfg2
GMI_WAIT	0x700031c8	atcfg2
GMI_WP_N	0x700031c0	atcfg3
GMI_WR_N	0x70003240	atcfg2
GPIO_PBB0	0x7000328c	gmecfg
GPIO_PBB3	0x70003298	gmecfg
GPIO_PBB4	0x7000329c	gmfcfg
GPIO_PBB5	0x700032a0	gmfcfg
GPIO_PBB6	0x700032a4	gmfcfg

Ball name	register address	pad control group
GPIO_PBB7	0x700032a8	gmfcfg
GPIO_PCC1	0x70003288	gmhcfg
GPIO_PCC2	0x700032ac	gmecfg
GPIO_PU0	0x70003184	dbgcfg
GPIO_PU1	0x70003188	dbgcfg
GPIO_PU2	0x7000318c	dbgcfg
GPIO_PU3	0x70003190	dbgcfg
GPIO_PU4	0x70003194	dbgcfg
GPIO_PU5	0x70003198	dbgcfg
GPIO_PU6	0x7000319c	dbgcfg
GPIO_PV0	0x70003040	uabcfg
GPIO_PV1	0x70003044	uabcfg
GPIO_PV2	0x70003060	uabcfg
GPIO_PV3	0x70003064	uabcfg
HDMI_CEC	0x700033e0	ceccfg
HDMI_INT	0x70003110	lcdcfg2
JTAG_RTCK	0x700032b0	dbgcfg
KB_COL0	0x700032fc	aocfg2
KB_COL1	0x70003300	aocfg2
KB_COL2	0x70003304	aocfg2
KB_COL3	0x70003308	aocfg2
KB_COL4	0x7000330c	aocfg2
KB_COL5	0x70003310	aocfg2
KB_COL6	0x70003314	aocfg2
KB_COL7	0x70003318	aocfg2
KB_ROW0	0x700032bc	aocfg1
KB_ROW1	0x700032c0	aocfg1
KB_ROW10	0x700032e4	aocfg2
KB_ROW11	0x700032e8	aocfg2
KB_ROW12	0x700032ec	aocfg2
KB_ROW13	0x700032f0	aocfg2
KB_ROW14	0x700032f4	aocfg2
KB_ROW15	0x700032f8	aocfg2
KB_ROW2	0x700032c4	aocfg1
KB_ROW3	0x700032c8	aocfg1
KB_ROW4	0x700032cc	aocfg1
KB_ROW5	0x700032d0	aocfg1
KB_ROW6	0x700032d4	aocfg1
KB_ROW7	0x700032d8	aocfg1
KB_ROW8	0x700032dc	aocfg2
KB_ROW9	0x700032e0	aocfg2
LCD_CS0_N	0x70003084	lcdcfg1
LCD_CS1_N	0x70003104	lcdcfg2
LCD_D0	0x700030a4	lcdcfg2
LCD_D1	0x700030a8	lcdcfg2
LCD_D10	0x700030cc	lcdcfg2
LCD_D11	0x700030d0	lcdcfg2
LCD_D12	0x700030d4	lcdcfg2
LCD_D13	0x700030d8	lcdcfg2
LCD_D14	0x700030dc	lcdcfg2
LCD_D15	0x700030e0	lcdcfg2
LCD_D16	0x700030e4	lcdcfg2
LCD_D17	0x700030e8	lcdcfg2
LCD_D18	0x700030ec	lcdcfg2
LCD_D19	0x700030f0	lcdcfg2
LCD_D2	0x700030ac	lcdcfg2
LCD_D20	0x700030f4	lcdcfg2
LCD_D21	0x700030f8	lcdcfg2
LCD_D22	0x700030fc	lcdcfg2
LCD_D23	0x70003100	lcdcfg2
LCD_D3	0x700030b0	lcdcfg2
LCD_D4	0x700030b4	lcdcfg2
LCD_D5	0x700030b8	lcdcfg2
LCD_D6	0x700030bc	lcdcfg2
LCD_D7	0x700030c0	lcdcfg2
LCD_D8	0x700030c4	lcdcfg2
LCD_D9	0x700030c8	lcdcfg2
LCD_DC0	0x70003088	lcdcfg1
LCD_DC1	0x7000310c	lcdcfg2
LCD_DE	0x70003098	lcdcfg2
LCD_HSYNC	0x7000309c	lcdcfg2
LCD_M1	0x70003108	lcdcfg2
LCD_PCLK	0x70003094	lcdcfg2
LCD_PWR0	0x70003090	lcdcfg2
LCD_PWR1	0x70003070	lcdcfg1
LCD_PWR2	0x70003074	lcdcfg1
LCD_SCK	0x7000308c	lcdcfg1
LCD_SDIN	0x70003078	lcdcfg1
LCD_SDOUT	0x7000307c	lcdcfg1
LCD_VSYNC	0x700030a0	lcdcfg2

Ball name	register address	pad control group
LCD_WR_N	0x70003080	lcdcfg1
OWR	0x70003334	owrcfg
PEX_L0_CLKREQ_N	0x700033c0	gpvcfg
PEX_L0_PRSNT_N	0x700033b8	gpvcfg
PEX_L0_RST_N	0x700033bc	gpvcfg
PEX_L1_CLKREQ_N	0x700033d0	gpvcfg
PEX_L1_PRSNT_N	0x700033c8	gpvcfg
PEX_L1_RST_N	0x700033cc	gpvcfg
PEX_L2_CLKREQ_N	0x700033dc	gpvcfg
PEX_L2_PRSNT_N	0x700033d4	gpvcfg
PEX_L2_RST_N	0x700033d8	gpvcfg
PEX_WAKE_N	0x700033c4	gpvcfg
PWR_I2C_SCL	0x700032b4	aocfg1
PWR_I2C_SDA	0x700032b8	aocfg1
PWR_INT_N	0x7000332c	aocfg2
SDMMC1_CLK	0x70003048	sdio1cfg
SDMMC1_CMD	0x7000304c	sdio1cfg
SDMMC1_DAT0	0x7000305c	sdio1cfg
SDMMC1_DAT1	0x70003058	sdio1cfg
SDMMC1_DAT2	0x70003054	sdio1cfg
SDMMC1_DAT3	0x70003050	sdio1cfg
SDMMC3_CLK	0x70003390	sdio3cfg
SDMMC3_CMD	0x70003394	sdio3cfg
SDMMC3_DAT0	0x70003398	sdio3cfg
SDMMC3_DAT1	0x7000339c	sdio3cfg
SDMMC3_DAT2	0x700033a0	sdio3cfg
SDMMC3_DAT3	0x700033a4	sdio3cfg
SDMMC3_DAT4	0x700033a8	sdio2cfg
SDMMC3_DAT5	0x700033ac	sdio2cfg
SDMMC3_DAT6	0x700033b0	sdio2cfg
SDMMC3_DAT7	0x700033b4	sdio2cfg
SDMMC4_CLK	0x70003258	gmccfg
SDMMC4_CMD	0x7000325c	gmdcfg
SDMMC4_DAT0	0x70003260	gmacfg
SDMMC4_DAT1	0x70003264	gmacfg
SDMMC4_DAT2	0x70003268	gmacfg
SDMMC4_DAT3	0x7000326c	gmacfg
SDMMC4_DAT4	0x70003270	gmbcfg
SDMMC4_DAT5	0x70003274	gmbcfg
SDMMC4_DAT6	0x70003278	gmbcfg

Ball name	register address	pad control group
SDMMC4_DAT7	0x7000327c	gmbcfg
SDMMC4_RST_N	0x70003280	gmacfg
SPDIF_IN	0x70003350	dap1cfg
SPDIF_OUT	0x70003354	dap1cfg
SPI1_CS0_N	0x70003380	spicfg
SPI1_MISO	0x70003384	spicfg
SPI1_MOSI	0x70003378	spicfg
SPI1_SCK	0x7000337c	spicfg
SPI2_CS0_N	0x70003370	spicfg
SPI2_CS1_N	0x70003388	spicfg
SPI2_CS2_N	0x7000338c	spicfg
SPI2_MISO	0x7000336c	spicfg
SPI2_MOSI	0x70003368	spicfg
SPI2_SCK	0x70003374	spicfg
SYS_CLK_REQ	0x70003320	aocfg2
UART2_CTS_N	0x70003170	uart2cfg
UART2_RTS_N	0x7000316c	uart2cfg
UART2_RXD	0x70003164	uart2cfg
UART2_TXD	0x70003168	uart2cfg
UART3_CTS_N	0x7000317c	uart3cfg
UART3_RTS_N	0x70003180	uart3cfg
UART3_RXD	0x70003178	uart3cfg
UART3_TXD	0x70003174	uart3cfg
ULPI_CLK	0x70003020	udacfg
ULPI_DATA0	0x70003000	uaacfg
ULPI_DATA1	0x70003004	uaacfg
ULPI_DATA2	0x70003008	uaacfg
ULPI_DATA3	0x7000300c	uaacfg
ULPI_DATA4	0x70003010	uabcfg
ULPI_DATA5	0x70003014	uabcfg
ULPI_DATA6	0x70003018	uabcfg
ULPI_DATA7	0x7000301c	uabcfg
ULPI_DIR	0x70003024	udacfg
ULPI_NXT	0x70003028	udacfg
ULPI_STP	0x7000302c	udacfg
VI_D0	0x70003124	vicfg1
VI_D1	0x70003128	vicfg1
VI_D10	0x7000314c	vicfg1
VI_D11	0x70003150	vicfg1
VI_D2	0x7000312c	vicfg1

Ball name	register address	pad control group
VI_D3	0x70003130	vicfg1
VI_D4	0x70003134	vicfg1
VI_D5	0x70003138	vicfg1
VI_D6	0x7000313c	vicfg1
VI_D7	0x70003140	vicfg1
VI_D8	0x70003144	vicfg1
VI_D9	0x70003148	vicfg1

Ball name	register address	pad control group
VI_HSYNC	0x70003160	vicfg1
VI_MCLK	0x70003158	csuscfg
VI_PCLK	0x70003154	vicfg1
VI_VSYNC	0x7000315c	vicfg1

Refer to the Excel spreadsheet, *Tegra 3 PinMux* (DA-05692) for a list of the SFIO functions supported by each MPIO on Tegra 3 devices. Each SFIO is listed in the table with the following format: <ControllerType><ControllerInstance><InterfaceLetter>-<Signal>

- **ControllerType** indicates the type of IO controller associated with this SFIO function
- **ControllerInstance** indicates the IO controller if the Tegra 3 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 which are omitted from the spreadsheet. Many of these are deprecated. Contact NVIDIA for more information.

8.6 Deep Sleep Behaviors

Deep Sleep is an ultra-low-power standby state in which a Tegra 3 device maintains much of its IO 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.

8.6.1 Deep Sleep Entry

The steps to enter deep sleep are as follows:

1. Software programs PMC to enter deep sleep
2. PMC latches much of the Tegra 3 device's I/O state, takes control of the I/O's, and resets the I/O controllers
3. PMC deasserts CORE_PWR_REQ
4. The PMU/PMIC powers down VDD_CORE
5. PMC continues driving the IO state that it has latched

> **Note:** For more information about deep sleep entry, refer to Section 2.5.3 in *NVIDIA Tegra 3 Mobile Quad-Core Processors Data Sheet* (DS-05160).

8.6.2 Deep Sleep Exit

1. PMC detects a wake event (for example, a rising edge on an appropriately configured MPIO)
2. PMC asserts CORE_PWR_REQ
3. The PMU/PMIC powers up VDD_CORE
4. PMC resets the logic within VDD_CORE

5. PMC yields control of some of the MPIO's to the I/O controllers (note that the controllers are in a power-on-reset state.)
6. The boot ROM executes and wakes up software
7. Software re-initializes the I/O controllers
8. When software requests, PMC yields control of the remaining MPIO's to the I/O controllers.

> **Note:** For more information about deep sleep exits, refer to Section 2.5.4 in *NVIDIA Tegra 3 Mobile Quad-Core Processors Data Sheet* (DS-05160).

8.6.3 Pad Capabilities During Deep Sleep

The MPIO pads do not all have identical behavior during deep sleep. They differ with regards to

- Output buffer behavior during deep sleep:
  - Does it maintain a programmable (0, 1, or tristate) constant value.
  - 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
  - Does it maintain its deep sleep state until software requests OR
  - Does PMC forcibly return the pad to its power-on-reset state before software is running?

Refer to Section 4.2.1, Multi-Purpose I/O Pads, in the document entitled *NVIDIA Tegra 3 Mobile Quad-Core Processors Data Sheet* (DS-05160) for a summary of the capabilities of each MPIO during deep sleep. See the columns labeled "Output Buffer", "Input Buffer", "Weak PU/PD", and "After Wake". The following sections describe in more detail how to interpret those columns.

8.6.3.1 OUTPUT Buffers During Deep Sleep

The output buffers of most MPIO pads are "configurable" during deep sleep. Immediately prior to entering deep sleep, PMC latches the state of the output buffers for such pads. If the pad was driving high (or low) prior to entering deep sleep, PMC ensures that it continues to drive high (or low) throughout deep sleep. If the pad was tri-stated during deep sleep, PMC ensures that it remains tri-stated 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 Tegra 3 device remains in deep sleep. For example, the keyboard controller may continue scanning the keypad matrix during deep sleep.

8.6.3.2 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 3 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.

8.6.3.3 Pull-Ups/-Downs During Deep Sleep

The weak pull-ups and pull-downs of most of the MPIO pads are “disabled” during deep sleep to reduce the Tegra 3 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, PMC ensures that it remains enabled throughout deep sleep. Similarly, if the pull-up (or pull-down) was disabled prior to entering deep sleep, PMC ensures that it remains enabled throughout deep sleep.

8.6.3.4 Behavior During Deep Sleep Exit

Many of the MPIO pads “reset” to their power-on-reset state as part of the deep sleep exit procedure. For such pads, it is advisable to ensure that their output state during deep sleep matches their power-on-reset state. Otherwise, the pad outputs might glitch during a wake from deep sleep.

The rest of the MPIO pads “hold” their deep sleep state until system-dependent software de-asserts the TRISTATE bit in the associated pinmux controller registers. This gives software the opportunity to re-initialize the pinmux and IO controllers without any glitches in the state of the MPIO pads.

8.7 GPIO Controller

The GPIO controller for Tegra 3 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 MPIO’s. Within a bank, the GPIO’s 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 E are in bank 1.

> **Note:** Refer to Section 4.2.1, Multi-Purpose I/O Pads, in *NVIDIA Tegra 3 Mobile Quad-Core Processors Data Sheet* (DS-05160) for a map of each of the Tegra 3 device MPIO pads to a particular GPIO port and bit. See the column labeled “GPIO.”

Refer to the section on “GPIO controller” in this document for more information on Tegra’s GPIO controller capabilities.

8.8 Programming Considerations

8.8.1 Controller Instances with Multiple Pin-outs

Several of the IO controller instances on a Tegra 3 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 SFIO’s. 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 four pins: ULPI_DATA0, SDMMC3_CLK, GPIO_PU0, and UART2_RTS_N. Depending on the value programmed into the corresponding pinmux registers, each of those four pins might toggle when UART A transmits data.

Similarly, RXD is an input signal to UART A. It is available on four pins: ULPI_DATA1, SDMMC3_CMD, GPIO_PU1, 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 four 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 three pins should be programmed to some other function.

8.8.2 Resume From Deep Sleep

As the Tegra 3 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.

8.8.3 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.

8.8.4 Drive Strengths

Using a weaker drive strength can reduce ringing which can reduce EMI and power dissipation.

8.8.5 VDDIO_VI, VDDIO_SDMMC4, PMC, and IO_RESET

Two power rails on the Tegra 3 devices, VDDIO_VI and VDDIO_SDMMC4, can operate at either 1.2V or 1.8V. VDDIO_VI and VDDIO_SDMMC4 supply power to LV (low voltage) MPIO pads.

To ensure the MPIOs on VDDIO_VI operate correctly at 1.2V, set bit 0 of APBDEV_PMC_DDR_PWR_0 to 0. To ensure the LV pads on VDDIO_SDMMC4 operate correctly at 1.2V, set bit 0 of APBDEV_PMC_DDR_PWR_0 to 1.

Bit 0 of APBDEV_PMC_DDR_PWR_0 **must** be set to 1 when VDD_VI is powered at 1.8V. Violating this constraint may permanently damage Tegra 3 device. Bit 1 of APBDEV_PMC_DDR_PWR_0 **must** be set to 1 when VDD_SDMMC4 is powered at 1.8V. Violating this constraint may permanently damage the Tegra 3 device.

The bits in APBDEV_PMC_DDR_PWR_0 reset to 1 during power-on-reset. This protects the chip from damage in case VDDIO_VI or VDDIO_SDMMC4 powers up at 1.8V. Early during boot, software should set APBDEV_PMC_DDR_PWR_0 to the appropriate values depending on the operating voltages of VDDIO_VI and VDDIO_SDMMC4. Once software has programmed APBDEV_PMC_DDR_PWR_0 correctly, software can clear the IO_RESET bits in the pinmux registers associated with LV pads.

APBDEV_PMC_DDR_PWR_0 is a register in the PMC. APBDEV_PMC_DDR_PWR_0 retains its state before, during, and after deep sleep. However, the pinmux registers (including their IO_RESET fields) lose their state during deep sleep. Software is responsible for reprogramming the pinmux registers upon resume from deep sleep.

Note: See the PMC section in this document for more information on APBDEV_PMC_DDR_PWR_0 and other PMC registers.

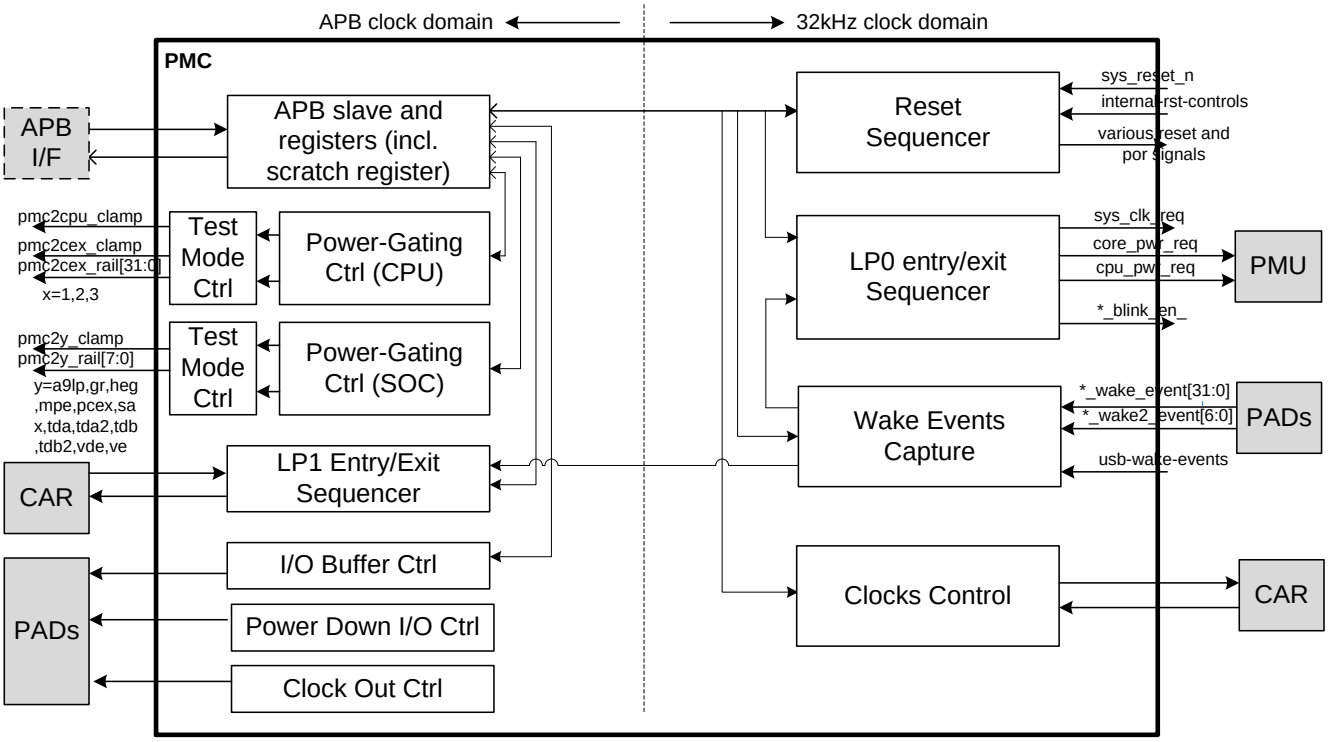
9.0 POWER MANAGEMENT CONTROLLER

Note: PRELIMINARY pending review

The Power Management Controller (PMC) block interacts with an external or Power Manager Unit (PMU). The PMC mostly controls the entry and exit of the system from different sleep modes. It provides power-gating controllers for SOC and CPU power-islands and also provides scratch storage to save some of the context during sleep modes (when CPU and/or SOC power rails are off). Additionally, PMC interacts with the external Power Manager Unit (PMU).

Sleep and deep sleep require specific logic to maintain some state and control the power domains, including signaling to the external PMU, to provide power to the main logic in the Tegra® 3 devices. All this logic is centralized in the PMC block.

Figure 9. PMC Block Diagram



Glossary

- Frozen boot: The RTC partition power transitions from OFF to ON
- Cold boot: The main partition power transitions from OFF to ON with no previous state available, SW must construct all state from scratch. Boot ROM is executed. DRAM is brought on-line.
- Warm boot: The main partition power transitions from OFF to ON with previous state available, SW checks for the preserved state integrity and restores the saved state from DRAM, which was in self-refresh prior to warm-boot. This is also called Deep Sleep wake-up or LP0 exit.
- LP0: Low Power 0 state in which, OS is suspended, DRAM is put in self-refresh, system state is saved in PMC + DRAM , VDD_SOC and VDD_CPU rails are powered off, and PMC is configured to monitor “LP0 wake events” which would trigger LP0 exit. This is also called Deep Sleep state.

- LP1: Low Power 1 state in which, OS is suspended, DRAM is put in self-refresh, VDD_CPU rail is powered off, Devices are power-gated, SoC clock domains are put into min frequency (32Khz) and Flow-Controller is configured to monitor “LP1 wake events” which would trigger LP1 exit.

9.1 External Interface

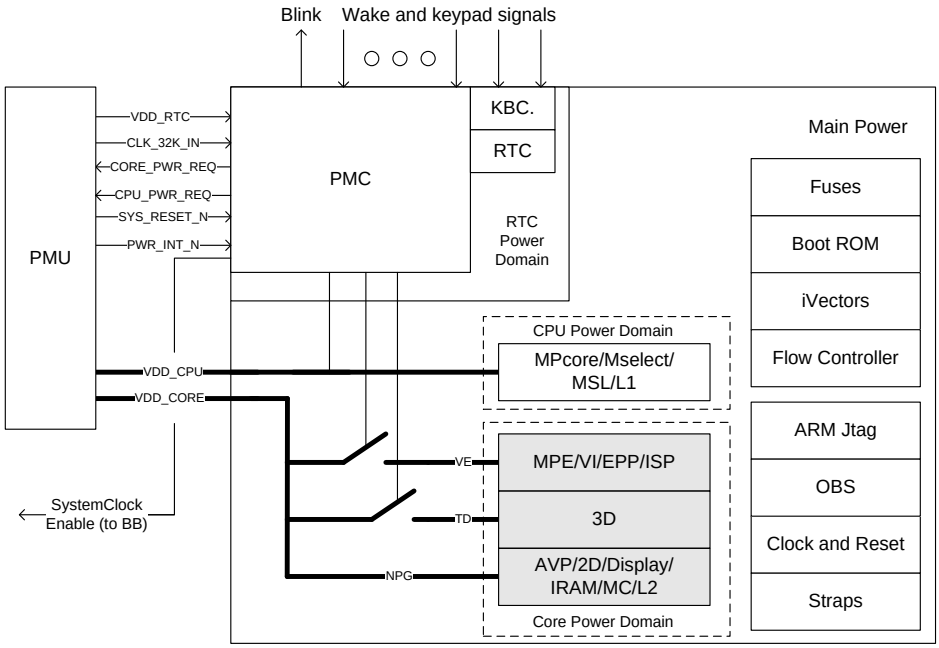
Table 27 PMC External Interface Signals

Signal Name	Type	Signal Description
SYS_RESET_N	In	Hardware reset, active low outside the chip
CORE_PWR_REQ	Out	Used to signal the external PMU device when to toggle the Main power state
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
CLK1_REQ	In	
CLK2_REQ	In	
CLK3_REQ	In	
CLK1_OUT	Out	
CLK2_OUT	Out	
CLK3_OUT	Out	

9.2 Functionality

PMC is a part of the RTC power domain. PMC manages the interface with the external PMU, 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 3 devices is shown below.

Figure 10. Power Domains in Tegra 3 Devices



The most important aspect of PMC is wake-up and reset logic. The following diagrams show the expected behavior of 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 3 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.

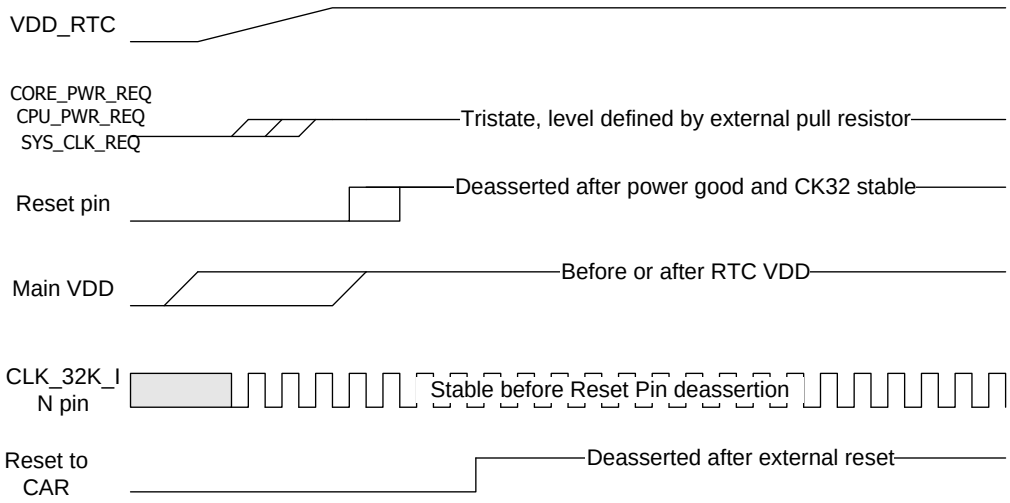
9.2.1 Frozen Boot Sequence

The Frozen Boot wake-up sequence happens when power has been de-asserted to the RTC domain:

1. PMU detects a turn-on condition and enables power to the VDD_RTC
2. Simultaneously to VDD_RTC enable or shortly afterwards, PMU enables power to VDD_CORE and optionally to VDD_CPU.
3. >1ms later PMU enables power to 1.8V I/O rails that need to be enabled for boot
4. >1ms later PMU enables power to 2.8V/3.3V I/O rails that need to be enabled for boot
5. PMU drives 32.768 kHz clock to Tegra 3 devices
6. >1ms later PMU deasserts reset to Tegra 3 devices, 32.768 kHz clock and all power rails must be stable and at their nominal value before reset deassertion

7. PMC block resets and deasserts reset to CAR block
8. Tegra 3 devices begins boot from internal ROM

Figure 11. Frozen Boot Sequence



9.2.2 The Deep Sleep Wake-up Sequence

The Deep Sleep wake-up sequence is the most complex for PMC. A large part of the complexity comes from the need to operate specific part of the logic and of the I/O using the VDD_RTC voltage and to avoid race conditions when going to sleep or waking up.

The corresponding I/O rail for a given wakeup pin needs to be powered on in order for the wakeup to function. The VDD_CORE and VDD_CPU power rails are turned off during DEEP SLEEP. Wakeup signals are routed from the pads directly to RTC for wakeup sequence to be initiated.

A full Deep Sleep / Wake-Up sequence proceeds according to the following steps:

Setting up Deep Sleep

1. SW disables unneeded logic blocks and peripherals not needed and disables their clocks.
2. SW programs PMC with the wake-up condition
3. SW programs scratch registers with information needed to restore state after wake-up
4. SW interacts with PMC to establish the idle conditions on the I/O
5. SW programs external SDRAM to enter self-refresh mode
6. SW programs the PMC to enter Deep Sleep
7. PMC asserts the clamping signals for all power gated islands
8. PMC deasserts power request (polarity previously programmed)
9. PMU removes VDD_CORE and VDD_CPU, Deep Sleep entered

Note: To Power-off VDD_CPU:

1. Assert resets to CPU via CLK_RST_CONTROLLER_RST_CPU_CMPLX_SET
2. Disable clock to CPU via CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0_CLK_ENB_CPU
3. Remove VDD_CPU

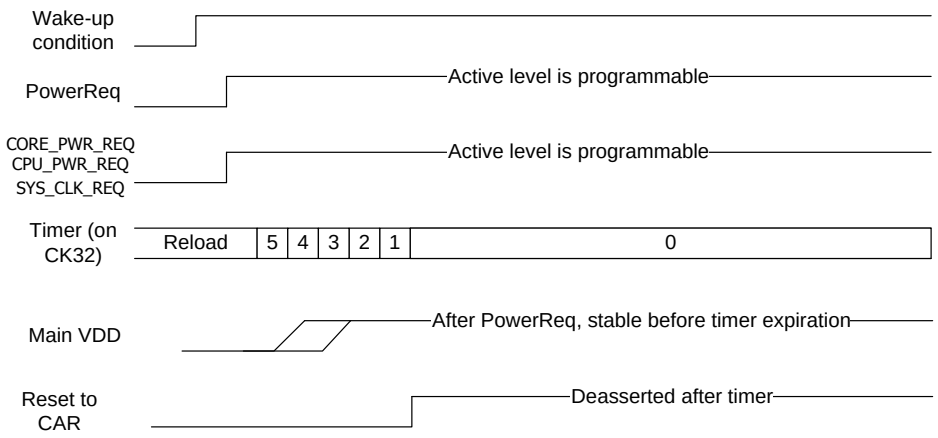
Deep Sleep Wakeup

1. PMC detects a wake-up condition and latches the condition into a register
2. PMC asserts the power request signal, asserts reset to CAR and starts timer (power good delay)
3. PMU restores VDD_CORE and optionally VDD_CPU
4. Timer expires, PMC releases the reset to main

Note: To Power-on VDD_CPU:

1. Restore VDD_CPU
2. Enable clock to CPU via CLK_RST_CONTROLLER_CLK_OUT_ENB_L_0_CLK_ENB_CPU
3. If using PLLX as CPU clock source is desired, configure/enable PLLX and switch to PLLX (this step can be done after step 4).
4. De-assert resets to CPU via CLK_RST_CONTROLLER_RST_CPU_CMPLX_CLR

Figure 12. Deep Sleep Wake-Up Sequence

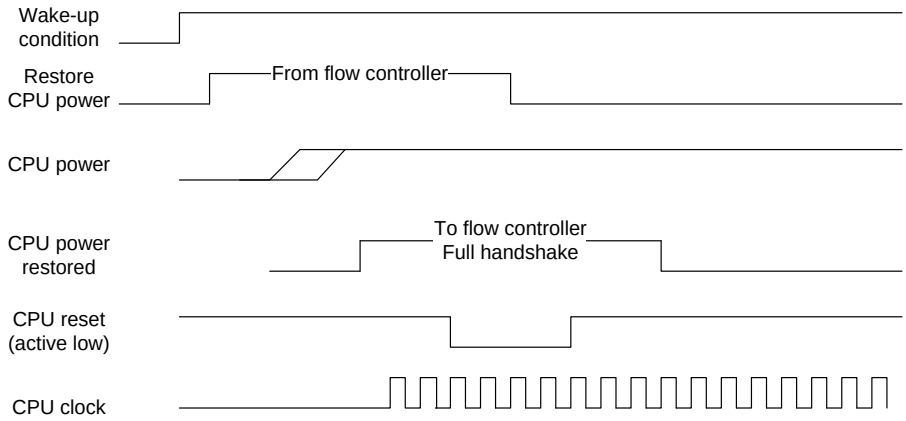


9.2.3 Suspend Mode Wake-up Sequence

The Suspend Mode wake-up sequence is simpler. PMC needs to wake-up CPU by removing the power gating:

1. SW configures flow controller with the Suspend Mode wake-up conditions.
2. Flow controller interacts with CPU to ensure that no transactions are in flight
3. Flow controller indicates to CAR to reset CPU
4. PMC clamps CPU signals and removes power from CPU
5. Flow controller detects Suspend mode wake-up condition
6. Flow controller sends a trigger event to PMC
7. PMC restores power to CPU
8. PMC sends an acknowledge to flow controller
9. Flow controller restores clock to CPU
10. CAR asserts reset to CPU
11. PMC removes clamping to the CPU
12. CAR deasserts reset to CPU

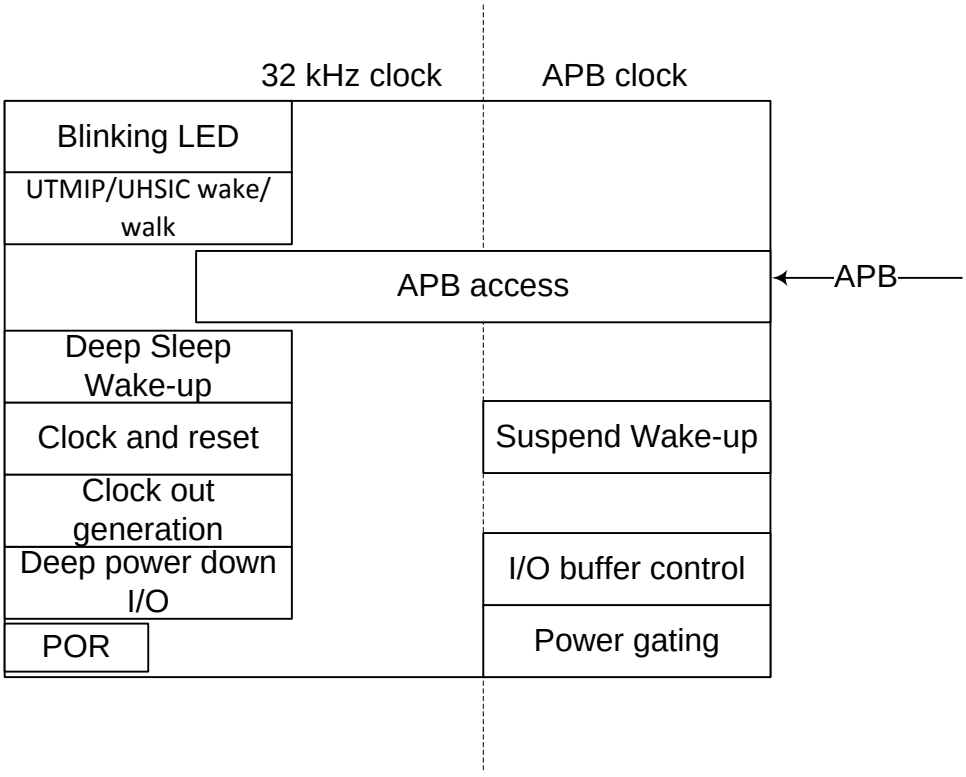
Figure 13. Suspend Mode Wake-Up Sequence



9.3 Reference Block Decomposition

The next figure shows a reference decomposition of PMC to simplify the description. The design uses a different functional partitioning.

Figure 14. Reference Block Decomposition Diagram



The blocks inside 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

9.3.1 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 has been described in more detail earlier in this section.

Note that entering Deep sleep mode should happen after an ordered shutdown of Tegra 3 devices. As part of the shutdown procedure, external interfaces must be placed in their idle condition. PMC provides the logic that ensures that interfaces maintain their idle state during Deep Sleep itself. The SW is responsible to sample the idle state by writing to the DPD sample bit before entering Deep Sleep mode.

SW must also ensure 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 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).

SW 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 Power Management Unit Specification.

The Deep Sleep mode entry/wake-up logic operates largely on the 32.768 kHz clock, so SW 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:

- Option1 (side-effect): First CPU power is gated, and as a result (“side-effect”) PMC performs an orderly LP0 entry. PMC HW also sets the PMC_DPD_ENABLE (automatically). Normally, this is the only option for LP0 entry.
- Option2 (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.

9.3.2 Deep Sleep Wake-Up Logic

The wake-up logic performs the following tasks:

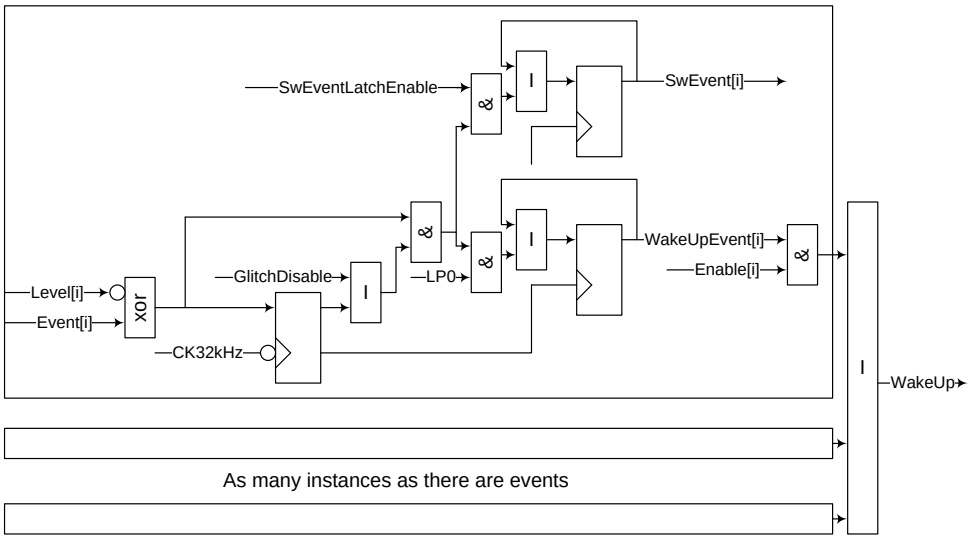
- Tracks the state of a programmable set of wake-up conditions
- Detects a wake-up condition
- Starts the sequence of action required by the Deep Sleep mode wake-up

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 don't 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 15. 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 PMU 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 SW 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 HW enters the Deep Sleep state, but after SW has instructed HW to enter Deep Sleep.
- Events that occur after the HW wake-up, but before SW has enabled the peripheral functions in the NPG that process the signals tied to the wake-up events, i.e. the GPIO event logic.

9.3.3 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 PMC can act upon it. For

pulse wake-up signals, PMC registers them even if they trigger during PMC_WAKE_DELAY window. However, PMC will act upon (registered) pulse event only after PMC_WAKE_DELAY window has expired.

PMC detects wake signals by using 32kHz logic. Therefore, in either case (level or pulse), a signal has to last long enough for PMC to recognize the change in signal. In addition, PMC provides 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 28. 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 us	1.5 period of 32.768 kHz clock = 45.755 us
1 (No glitch reject)	0 period of 32.768 kHz clock = 0 us	1 period of 32.768 kHz clock = 30.50 us

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.

Note that the logic doesn't contain any formal synchronization stage and so theoretically metastability cannot be precluded at this time. However the MTBF linked to metastability will be insignificant given the very low active duty of the wake-up circuitry and the low frequency of the sample clock (32.768 kHz). Adding an explicit synchronization stage increases the latency of the circuit.

9.3.4 Suspend Wake-up Logic

The only Suspend mode wake-up specific logic in PMC is related to the power gating logic as described previously.

9.3.5 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 definition). The CPU power gated domain has an extra trigger coming from the flow controller. PMC and flow controller provides a full handshake, with PMC providing an acknowledge back to 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 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 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:

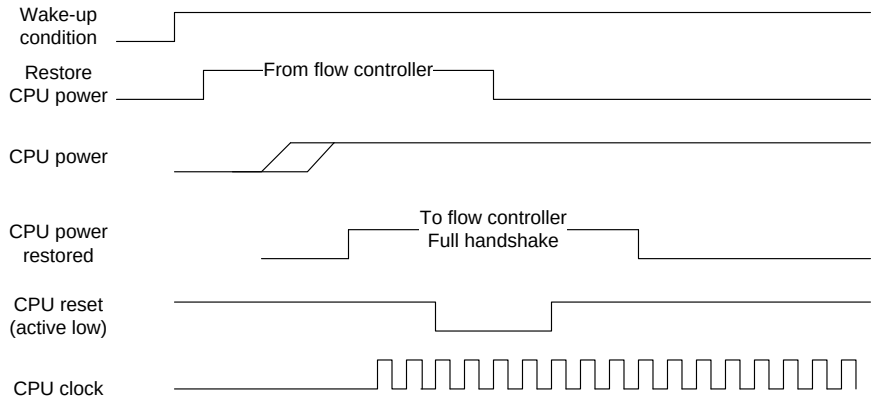
1. The CPU decides to enter either Deep Sleep or Suspend mode.
2. In both cases, the power to the CPU must be gated off
3. CPU does any required clean up job. This is especially important for Deep Sleep (all interfaces in idle, save context, etc.)
4. The CPU informs flow controller it wants to remove its power, with two variants:

5. For Suspend mode, the CPU also defines the set of events that will wake up the CPU (the flow controller is not power gated together with the CPU)
6. For Deep Sleep, write to sample register preserves I/O state to be driven when the chip enters Deep Sleep mode
7. For Deep Sleep, CPU must define in PMC the set of wake-up events for the whole Tegra 3 devices, and set the DPD trigger enable bit
8. CPU enters an endless loop
9. The flow controller asserts reset to the CPU
10. The flow controller informs the PMC that CPU power gating OFF is requested
11. The PMC power gates the CPU OFF and then informs the flow controller
12. If the DPD trigger enable was set, PMC also performs an Deep Sleep entry, this will result after awhile 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 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. CPU checks different status registers to establish this was in fact a Suspend mode wake-up.

Figure 16. Power Gating Logic



9.3.6 I/O Buffer Control

The logic for 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 SW sequencing:

1. SW 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 SW 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. SW 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. SW triggers the Deep Sleep mode, killing itself
5. HW detects the Deep Sleep wake-up condition
6. SW wakes-up and restores the state

7. SW clears the DPD sample bit
8. SW clears the DPD enable bit
9. I/O control can be sequenced back to the controlling block once SW has initialized the core block to drive outputs to the same idle state they maintained before entering DPD.

PMC also contains a SW 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 SW before shutting off the corresponding I/O power. Reversely, each bit should be reset after transitioning back the IO voltage to On (and before enabling the set of interfaces using the corresponding I/O rail).

PMC also contains two registers related to 3.3V I/O support. Specific power detector cells indicate per I/O voltage domain if their supply voltage is low or high. This signal must be reflected in a corresponding signal of the I/O pads part of that I/O voltage domain. The detector cells consume significant power, so they cannot be left on all the time. PMC contains a register enabling the detector cells. The output of the detector cell can then be sampled and written in another register linked to the control signal going to each I/O voltage domain.

9.3.7 Power Down I/O Logic

The new deep power-down domains are created for pad control. They allow enabling deep power down mode on single power rail or single pad brick without entering LP0. Unlike in LP0 mode, the function is implemented in ABP clock domain; therefore, some extra controls are needed to guarantee proper timing on the pads. The DPD domain groups added 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 single DPD domain has simplified steps of LP0 entry. To set a domain in DPD mode:

- Program interface to drive idle state out
- Set sample register (PMC_DPD_SAMPLE)
- Set fields corresponding to dpd domains to 1 in PMC_IO_DPD_REQ register or PMC_IO_DPD2_REQ register and CODE to DPD_ON
  - Note, Fields set to 0 will mean no status change
  - Note, IO_DPD_STATUS and IO_DPD2_STATUS are read only registers which display status of each rail. PMC_DPD_SAMPLE register must be reset before next sample operation.
- Set PMC_SEL_DPD_TIM to value corresponding to minimum 80 ns (this allows for proper timing sequence between sel_dpd, e_dpd)
- Set interface to drive idle state

- Set fields corresponding to dpd domains to 1 in PMC_IO_DPD_REQ register or PMC_IO_DPD2_REQ register and CODE to DPD_OFF
- Set tristate field in miscellaneous register to disable **for all pads in the set of affected domains** (this will deassert sel_dpd in mini pad macros). Some bricks do not have mini_pad_macros. For those, this step will be skipped.
  - Note, fields set to 0 will mean no status change

The following DPD domains do not have mini pad macros or do not drive 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 domain in DPD mode:

- Set fields corresponding to dpd domains to 1 in MC_IO_DPD_REQ register or MC_IO_DPD2_REQ register and CODE to DPD_ON
  - Note, fields set to 0 will mean no status change

To reset DPD mode for the DPD domain:

- Set PMC_SEL_DPD_TIM to value corresponding to minimum 80 ns (this allows for proper timing sequence between sel_dpd, e_dpd)
- Set fields corresponding to dpd domains to 1 in PMC_IO_DPD_REQ register or PMC_IO_DPD2_REQ register and CODE to DPD_OFF
  - Note, fields set to 0 will mean no status change

> **Note:** In 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. Sequence of multiple DPD sets require keeping all interfaces already in DPD mode, and driven as idle. *We can either add 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, POP_CLK. So to fully turn off vi, we need to turn off all three groups.

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

9.3.8 Clock Control Logic

The Tegra 3 device has a new functionality to allow oscillator to run in LP0 mode. It is combined with the ability to output clock (CLK_OUT) which can be extended outside LP0. Based on register setup, PMC can take over the control of Oscillator during DPD mode. When enabled, PMC takes over oscillator pad control by setting oscfi_d, oscfi_s, oscfi_en. See PMC_OSC_EDPD_OVER and PMC_CLK_OUT_CNTRL registers for register description.

There are two functions embedded in this feature:

- Programmable option of keeping oscillator ON during DPD mode
- For low cost systems, ability to output clock on dap_mclk1, dap_mclk2, clk3_out pins.

Programmable option of keeping oscillator ON during DPD mode

When programmed in PMC_OSC_EDPD_OVER register to take over the control of oscillator (OSC_CTRL_SELECT bit), PMC starts controlling all inputs to oscillator pad, including enable. However, setting EN bit to 0 (disabling oscillator) would be allowed only during deep power down mode, with automatic enabling on leaving LP0 (this is to avoid system lockup). Based on PMC_OSC_EDPD_OVER setup oscillator can be controlled as follows:

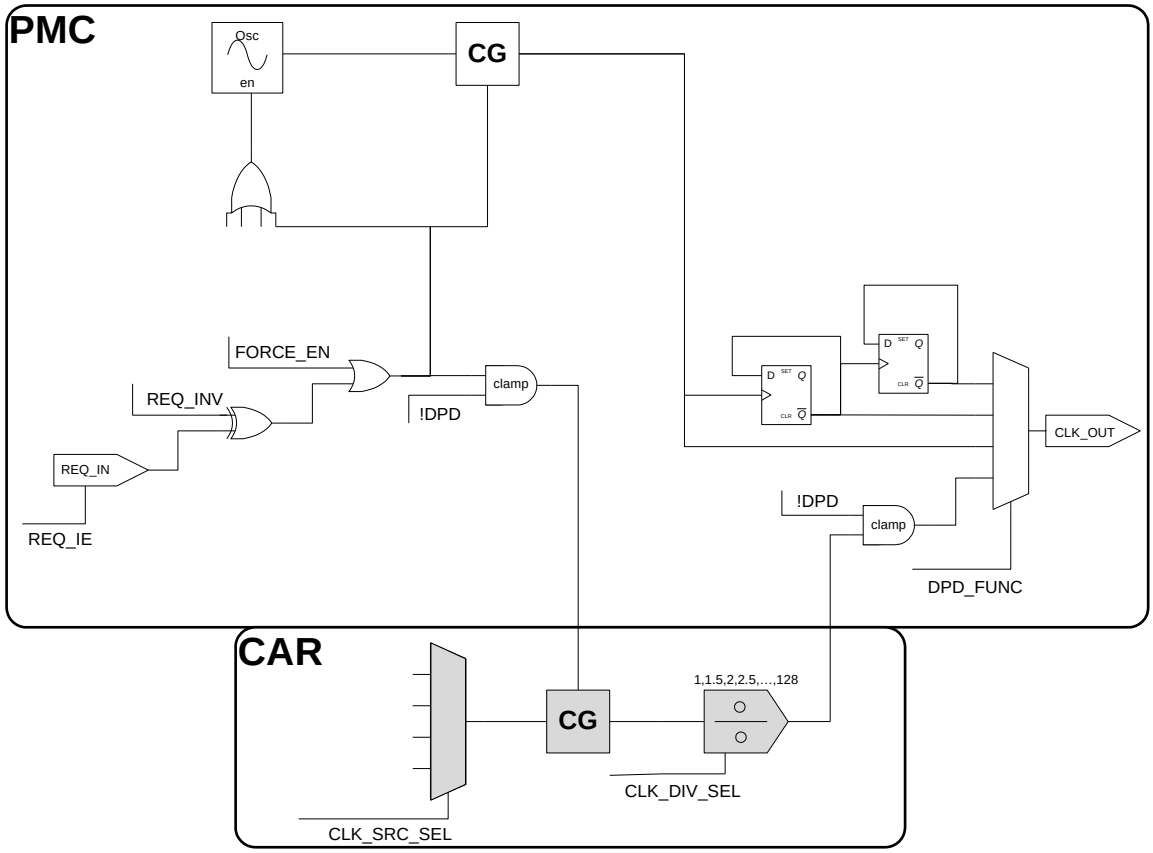
- Completely controlled by CAR, entering deep power down mode on LP0 (shutting down oscillator)
- Controlled by PMC, entering deep power down mode on LP0
- Controlled by PMC, not entering deep power down mode, but oscillator output is disabled (used to accelerate warm boot time)
- Controlled by PMC, not entering deep power down mode, oscillator output enabled – used for clk 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 corresponding clock outputs: dap_mclk1, dap_mclk2, clk3_out. When such a request is decoded, 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 clock request and for source of clock output itself.

Request can be decoded as active high, low, or 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 tri-stated. Once request is accepted, clock source could be CAR output, or 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 17. CLK_OUT paths



9.3.9 Clamping

This is related to power gating, when a partition is OFF. All signals leaving that partition are clamped to their idle value, normally 0, as most signals are active high.

Applying the clamping presents no specific problem, as all blocks into the partition being powered OFF are assumed to be in reset prior to the power gate transition.

Removing the clamping is more complex because Tegra 3 devices use a synchronous reset strategy, so the correct sequence is the following:

- Restore power, reset and clamping 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 SW. Writing to a trigger register removes the clamping for the identified set of partitions. This doesn't work for Suspend mode exit, where the CPU must be restarted by HW only.

In Suspend mode exit, there is an interlock between the CAR (Clock And Reset block) and PMC, so that the correct sequence is performed. CAR indicates to 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. PMC removes the CPU clamping as soon as it receives the request to do so. This is similar to the interlock between PMC and flow controller, except that this is not a full handshake.

9.3.10 Scratch Registers

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 57 PMC scratch registers, 8 secure registers, and 5 bondout registers in all. To save area, secure scratch registers 4-7, scratch registers 1-55, and 5 bondout registers are implemented in RAM. Due to requirement of being backward compatible with Tegra 2 address space, none of the three types has continuous address space.

9.3.11 Blinking

Tegra 3 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.

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.

9.3.12 Pin Muxing

Some of the signals controlled by 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 pin muxing. This is not directly a responsibility of PMC but the correct pin muxing must be present. 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 RTC logic. The ownership bit controls a final stage of muxing on top of the more general pin muxing stage.

As an important special case, the system clock enable is one of the Deep Sleep wake-up event pin. Obviously usage as a wake-up event pin is incompatible with usage as a system clock enable pin.

9.3.13 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 can't 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.

9.4 Sensor Reset

PMC monitors temperature sensor signal. When the signal becomes active, 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 whole chip automatically causes CPU1-3 power gating and request for turning down cpu_power.

Power gating is simplified version of regular power gating function. In first cycle clamping is enabled. In next cycle CPU1-3 is 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 thermal sensor event has happened. If PMC is entering LP0 mode while temp sensor triggers, the reset as a direct result of sensor active will not happen (to avoid destruction of PMC state). However, process of entering LP0 causes reset a few cycles later. CPU1-3 will be shutdown in each situation.

9.5 Register Definition for PMC

To speed up operation, this 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 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 which 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 500ms** when rail is off may reduce the lifetime of the pad. Setting it to 1 when 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_SATUS
- 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 defunct
- 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.

9.6 PMC Registers

9.6.1 APBDEV_PMC_CNTRL_0

Base offset in APB device space.

This register controls clock to KBC, RTC, reset to car, KBC, RTC. It also manages polarity and output enable of sys_clk_req pin and core_pwr_req pin at reset both are disabled since PMU properties are unknown.

Auxiliary functions - enables blinking functions, and software wake-up latching.

PMC Control Register

Offset: 000h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
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
15	0x0	CPUPWRREQ_POLARITY: Inverts power request polarity 0 = NORMAL 1 = INVERT
14	0x0	SIDE_EFFECT_LP0: when set causes side effect of entering LP0 after powering down CPU 0 = DISABLE 1 = ENABLE
13	0x0	AOINIT: AO (Always On or RTC domain) initialized purely SW 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 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

Bit	Reset	Description
7	0x0	BLINK_EN: Enables blinking counter and blink output works only if BLINK field in DPD_PADS_ORIDE is set to 1. 0 = DISABLE 1 = ENABLE
6	0x0	GLITCHDET_DIS: Disable detecting glitch 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 32khz clock, will be causing wakeup 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: now resets everything but scratch 0 and reset status 0 = DISABLE 1 = ENABLE
3	0x0	KBC_RST: Software reset to KBC 0 = DISABLE 1 = ENABLE
2	0x0	RTC_RST: Software reset to RTC. Defunct for Tegra 3 0 = DISABLE 1 = ENABLE
1	0x0	RTC_CLK_DIS: Disable 32KHz clock to RTC 0 = DISABLE 1 = ENABLE
0	0x0	KBC_CLK_DIS: Disable 32KHz clock to KBC 0 = DISABLE 1 = ENABLE

9.6.2 APBDEV_PMC_SEC_DISABLE_0

On separate reset (same as car), disables access (read/write to secure scratch registers). Once write is disabled, secure registers cannot be written until power on reset or reset on exiting Deep Sleep.

Once read is disabled, read from scratch registers will return 0 until power on reset or reset on exiting Deep Sleep mode.

Single register can be disabled for read/write, bit 0,1 have overwrite function

Secure Register Disable

Offset: 004h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
19	0x0	READ7: disable read from secure register 7 0 = OFF 1 = ON
18	0x0	WRITE7: disable write to secure register 7 0 = OFF 1 = ON

Bit	Reset	Description
17	0x0	READ6: disable read from secure register 6 0 = OFF 1 = ON
16	0x0	WRITE6: disable write to secure register 6 0 = OFF 1 = ON
15	0x0	READ5: disable read from secure register 5 0 = OFF 1 = ON
14	0x0	WRITE5: disable write to secure register 5 0 = OFF 1 = ON
13	0x0	READ4: disable read from secure register 4 0 = OFF 1 = ON
12	0x0	WRITE4: disable write to secure register 4 0 = OFF 1 = ON
11	0x0	READ3: disable read from secure register 3 0 = OFF 1 = ON
10	0x0	WRITE3: disable write to secure register 3 0 = OFF 1 = ON
9	0x0	READ2: disable read from secure register 2 0 = OFF 1 = ON
8	0x0	WRITE2: disable write to secure register 2 0 = OFF 1 = ON
7	0x0	READ1: disable read from secure register 1 0 = OFF 1 = ON
6	0x0	WRITE1: disable write to secure register 1 0 = OFF 1 = ON
5	0x0	READ0: disable read from secure register 0 0 = OFF 1 = ON
4	0x0	WRITE0: disable write to secure register 0 0 = OFF 1 = ON
3:2	0x0	NOT_USED: legacy - redundant - not used
1	0x0	READ: disable read from secure registers - all 0 = OFF 1 = ON
0	0x0	WRITE: disable write to secure registers - all 0 = OFF 1 = ON

9.6.3 APBDEV_PMC_PMC_SWRST_0

Reset only by por cell and sets itself back to inactive after 2 clock cycles. Only for emergency debugging purposes and not to be used in any functional mode.

Note: Defunct for Tegra 3 devices. Kept only for binary code compatibility only. Will not do anything

Register write which resets PMC only

Offset: 008h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	RST: software reset to pmc only 0 = DISABLE 1 = ENABLE

9.6.4 APBDEV_PMC_WAKE_MASK_0

Masks which event can cause wake-up from Deep Sleep mode. Has to be setup before entering Deep Sleep. Works in conjunction with WAKE_LVL register.

Only enabled events at the proper wake_lvl will cause exit from Deep Sleep mode.

Current pin assignments wake_mask, wake_status, sw_wake_status and wake levels:

- WAKE_STATUS[0] = ulpi_data4
- WAKE_STATUS[1] = gp3_pv[1]
- WAKE_STATUS[2] = dvi_d3
- WAKE_STATUS[3] = sdmmc3_dat1
- WAKE_STATUS[4] = hdmi_int
- WAKE_STATUS[5] = sdmmc4_dat6
- WAKE_STATUS[6] = gp3_pu[5]
- WAKE_STATUS[7] = gp3_pu[6]
- WAKE_STATUS[8] = gmi_wp_n
- WAKE_STATUS[9] = gp3_ps[2]
- WAKE_STATUS[10] = gmi_ad21
- WAKE_STATUS[11] = spi2_cs2
- WAKE_STATUS[12] = spi2_cs1
- WAKE_STATUS[13] = sdmmc1_dat1
- WAKE_STATUS[14] = pe_wake_1
- WAKE_STATUS[15] = gmi_cs1_n
- WAKE_STATUS[16] = rtc_irq
- WAKE_STATUS[17] = kbc_interrupt
- WAKE_STATUS[18] = pwr_int
- WAKE_STATUS[19] = usb_vbus_wakeup[0]
- WAKE_STATUS[20] = usb_vbus_wakeup[1]
- WAKE_STATUS[21] = usb_iddig[0]
- WAKE_STATUS[22] = usb_iddig[1]

- WAKE_STATUS[23] = gmi_iordy
- WAKE_STATUS[24] = gp3_pv[0]
- WAKE_STATUS[25] = gp3_ps[4]
- WAKE_STATUS[26] = gp3_ps[5]
- WAKE_STATUS[27] = gp3_ps[0]
- WAKE_STATUS[28] = gp3_ps[6]
- WAKE_STATUS[29] = gp3_ps[7]
- WAKE_STATUS[30] = dap1_dout
- WAKE_STATUS[31] = 0

PMC Wake-up Event Mask

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000000000000

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

9.6.5 APBDEV_PMC_WAKE_LVL_0

This register sets level which is active level for wake event. Will cause exit from Deep Sleep mode if input signal level matches level set in this register and WAKE_MASK is set for the event to 1

PMC Wake Level

Offset: 010h | Read/Write: R/W | Reset: 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 - now permanently tied to bit 18 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
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

9.6.6 APBDEV_PMC_WAKE_STATUS_0

This register stores status of the wake events. Event will be set if level matches and is not masked.

Write will reset wake event set.

PMC Wake Status

Offset: 014h | Read/Write: R/W | Reset: 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

Bit	Reset	Description
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

9.6.7 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 PMC_CNRL register bit LATCHWAKE_EN.

Latching will stop at 1 to 0 transition on this bit. Event will be set if level matches. Masking is not affecting this register. Write will reset wake events set.

PMC Software Wake Status

Offset: 018h | Read/Write: R/W | Reset: 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

9.6.8 APBDEV_PMC_DPD_PADS_ORIDE_0

This register enables overrides idle state values and deep power down mode with values driven by KBC (keyboard) or PMC (SYS_CLK_REQ, LED_BLINK).

If bit is set to 1, particular I/O will drive core value, not DPD idle value (dpd_io_0/1). Pad will not be put in deep power down mode. It will not drive data from mini pad macros stored during sample cycle, but direct data from KBC or PMC.

> **Note:** This register has to be set before entering Deep Sleep mode.

DPD Pads Override

Offset: 01ch | Read/Write: R/W | Reset: 0b00001000000000000000000000

Bit	Reset	Description
25	0x0	KBC_ROW10: override dpd idle state with row 10 output 0 = DISABLE 1 = ENABLE
24	0x0	KBC_ROW9: override dpd idle state with row 9 output 0 = DISABLE 1 = ENABLE
23	0x0	KBC_ROW8: override dpd idle state with row 8 output 0 = DISABLE 1 = ENABLE
22	0x0	KBC_ROW7: override dpd idle state with row 7 output 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: override dpd idle state with row 6 output 0 = DISABLE 1 = ENABLE
18	0x0	KBC_ROW5: override dpd idle state with row 5 output 0 = DISABLE 1 = ENABLE
17	0x0	KBC_ROW4: override dpd idle state with row 4 output 0 = DISABLE 1 = ENABLE
16	0x0	KBC_ROW3: override dpd idle state with row 3 output 0 = DISABLE 1 = ENABLE
15	0x0	KBC_ROW2: override dpd idle state with row 2 output 0 = DISABLE 1 = ENABLE
14	0x0	KBC_ROW1: override dpd idle state with row 1 output 0 = DISABLE 1 = ENABLE
13	0x0	KBC_ROW0: override dpd idle state with row 0 output 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
12	0x0	KBC_COL12: override dpd idle state with column 12 output 0 = DISABLE 1 = ENABLE
11	0x0	KBC_COL11: override dpd idle state with column 11 output 0 = DISABLE 1 = ENABLE
10	0x0	KBC_COL10: override dpd idle state with column 10 output 0 = DISABLE 1 = ENABLE
9	0x0	KBC_COL9: override dpd idle state with column 9 output 0 = DISABLE 1 = ENABLE
8	0x0	KBC_COL8: override dpd idle state with column 8 output 0 = DISABLE 1 = ENABLE
7	0x0	KBC_COL7: override dpd idle state with column 7 output 0 = DISABLE 1 = ENABLE
6	0x0	KBC_COL6: override dpd idle state with column 6 output 0 = DISABLE 1 = ENABLE
5	0x0	KBC_COL5: override dpd idle state with column 5 output 0 = DISABLE 1 = ENABLE
4	0x0	KBC_COL4: override dpd idle state with column 4 output 0 = DISABLE 1 = ENABLE
3	0x0	KBC_COL3: override dpd idle state with column 3 output 0 = DISABLE 1 = ENABLE
2	0x0	KBC_COL2: override dpd idle state with column 2 output 0 = DISABLE 1 = ENABLE
1	0x0	KBC_COL1: override dpd idle state with column 1 output 0 = DISABLE 1 = ENABLE
0	0x0	KBC_COL0: override dpd idle state with column 0 output 0 = DISABLE 1 = ENABLE

9.6.9 APBDEV_PMC_DPD_SAMPLE_0

Setting this register will trigger sampling pads data and direction in which pad will be driven during Deep Sleep mode.

Before writing to this register, all interfaces going to pads must be set to ideal "idle" mode, which is expected to be driven by pads when chip enters Deep Sleep.

DPS Power Down Sample has to precede Deep Power Down Enable write. DPD Sample should not be deasserted until transition from Deep Sleep to WB0.

Deep Power Down Sample

Offset: 020h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	ON: will set sampling of pads value 0 = DISABLE 1 = ENABLE

9.6.10 APBDEV_PMC_DPD_ENABLE_0

Setting this register will trigger entering Deep Sleep state. Must be preceded by DPD_SAMPLE write.

Entering deep power down mode (Deep Sleep) will trigger shut down core power request and shut down system clock request. PLLs, I/Os will enter power deep down mode. All signals going from core power to RTC and pads will be cut off (clamped)

If none of the wake-up events is set, only power on reset can enable back access to the chip.

After serving wake-up event is completed, the register should be set back to 0 to complete Deep Sleep cycle.

Deed Power Down Enable

Offset: 024h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	ON: will set sampling of pads value 0 = DISABLE 1 = ENABLE

9.6.11 APBDEV_PMC_PWRGATE_TIMER_OFF_0

Specifies 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 powered partition. This register should be set before write to Power Gate Toggle. Shared between all power partitions.

Power Gate Timer Off Register

Offset: 028h | Read/Write: R/W | Reset: 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
11:8	0x5	RAIL2: timer value for rail 2
7:4	0x3	RAIL1: timer value for rail 1

Bit	Reset	Description
3:0	0x1	RAIL0: timer value for rail 0

9.6.12 APBDEV_PMC_PWRGATE_TIMER_ON_0

Specifies number of APB cycles after which the rail line goes on. Shared between all power partitions. Has to be reloaded before each power gate on/off command.

Note: For Tegra 3 this register is being kept for backward code compatibility, timer is based on PWRGATE_TIMER_OFF only

Power Gate Timer On Register

Offset: 02ch | Read/Write: R/W | Reset: 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
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

9.6.13 APBDEV_PMC_PWRGATE_TOGGLE_0

Write to this register will turn power on/off on to specified power gated partition. Only one partition will be turned on/off at the time.

PWRGATE_STATUS should be read to determine state of the partition before writing to this register.

Turning partition off will cause automatic clamping of all signals generated by the power gated partition being turned off. Before partition is turned off, all clocks to the partition should be stopped and reset to the partition should be asserted.

Turning partition on will not remove clamping. Clamping will be removed only after REMOVE_CLAMPING_CMD write.

User must allow minimum 20 apb clock cycles between consecutive partition Power Gate Toggle requests.

Power Gate Toggle

Offset: 030h | Read/Write: R/W | Reset: 0b0xxxx0000

Bit	Reset	Description
8	0x0	START: start power down/up 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
3:0	0x0	PARTID: id of partition to be toggled 0 = CP 1 = TD 2 = Video Encode 4 = Video Decode 3 = PCI Express 5 = L2 Cache 6 = MPEG Encode 7 = HEG 8 = SAX 9 = CE1 10 = CE2 11 = CE3 12 = A9LP 13 = TD2

9.6.14 APBDEV_PMC_REMOVE_CLAMPING_CMD_0

This is a bit map with one bit per power partition controller by 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 and clamping of the partition will continue

The bit is automatically reset to 0b when the clamping has been removed. SW is responsible to write to this register at a correct time, that is, when the clocks are started but the blocks in that partition are still held in reset.

Remove Clamping

Offset: 034h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	TD2: remove clamping to TD2 0 = DISABLE 1 = ENABLE
12	0x0	A9LP: remove clamping to A9LP 0 = DISABLE 1 = ENABLE
11	0x0	CE3: remove clamping to CE3 0 = DISABLE 1 = ENABLE
10	0x0	CE2: remove clamping to CE2 0 = DISABLE 1 = ENABLE
9	0x0	CE1: remove clamping to CE1 0 = DISABLE 1 = ENABLE
8	0x0	SAX: remove clamping to SAX 0 = DISABLE 1 = ENABLE
7	0x0	HEG: remove clamping to HEG 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
6	0x0	MPE: remove clamping to MPE_CACHE 0 = DISABLE 1 = ENABLE
5	0x0	L2C: remove clamping to L2_CACHE 0 = DISABLE 1 = ENABLE
4	0x0	PCX: remove clamping to PCX 0 = DISABLE 1 = ENABLE
3	0x0	VDE: remove clamping to VDE 0 = DISABLE 1 = ENABLE
2	0x0	VE: remove clamping to VE 0 = DISABLE 1 = ENABLE
1	0x0	TD: remove clamping to TD 0 = DISABLE 1 = ENABLE
0	0x0	CPU: remove clamping to CPU 0 = DISABLE 1 = ENABLE

9.6.15 APBDEV_PMC_PWRGATE_STATUS_0

Displays status of the power partitions. This register should be read before writing to PWRGATE_TOGGLE. Read only register.

Power Gate Status

Offset: 038h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

Bit	Reset	Description
13	X	TD2: status of TD2 partition 0 = OFF 1 = ON
12	X	A9LP: status of A9LP 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

Bit	Reset	Description
8	X	SAX: status of SAX 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
4	X	VDE: status of VDE partition 0 = OFF 1 = ON
3	X	PCX: status of PCX 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	CPU: status of CPU partition 0 = OFF 1 = ON

9.6.16 APBDEV_PMC_PWRGOOD_TIMER_0

Programs length of wakeup reset, asserted after wakeup from lp0. The register should be set before entering lp0 mode.

Programmed values depend on the properties of the power management unit.

Timer value x 30.51 us = reset pulse length

Power Good Timer

Offset: 03ch | Read/Write: R/W | Reset: 0b0000000001111111

Bit	Reset	Description
15:8	0x0	DATA: xtal timer * 32
7:0	0x7f	PMU_TIM: pmu timer * 32

9.6.17 APBDEV_PMC_BLINK_TIMER_0

Will output value to pad only if extra secure scratch 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 us
- Off time DATA_OFF<<2 x 30.51 us.

Blinker Timer, for External Blinker

Offset: 040h | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:16	0xffff	DATA_OFF: time off
15	0x1	FORCE_BLINK: if 0 32khz clock
14:0	0x7fff	DATA_ON: time on

9.6.18 APBDEV_PMC_NO_IOPOWER_0

Controls power rails for each I/O level. Can only be set when specific power rail is off. Setting it otherwise will damage the pad.

No I/O Power Register

Offset: 044h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
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
11	0x0	PEX_CNTRL: rail pex_cntrl 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
8	0x0	SD: rail sd I/Os 0 = DISABLE 1 = ENABLE
7	0x0	MEM: rail mem I/Os 0 = DISABLE 1 = ENABLE
6	0x0	LCD: rail LCD I/Os 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	0x0	AUDIO: rail I2S I/Os 0 = DISABLE 1 = ENABLE
4	0x0	VI: rail 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
0	0x0	SYS: rail AO I/Os 0 = DISABLE 1 = ENABLE

9.6.19 APBDEV_PMC_PWR_DET_0

Active high, sets power detection, only for nine power rails - MIPI doesn't 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 us delay (in APB clock cycles)
- write 1 to PWR_DET_LATCH

For turning PWR_DET off:

- write 0 to PWR_DET_LATCH
- no delay necessary
- write 0 to PWR_DET

Power Detect

Offset: 048h | Read/Write: R/W | Reset: 0b11111111111111

Bit	Reset	Description
13	0x1	SDMMC3: rail SDMMC3 I/Os 0 = ENABLE 1 = DISABLE
12	0x1	SDMMC1: rail SDMMC1 I/Os 0 = ENABLE 1 = DISABLE
11	0x1	PEX_CNTRL: rail pex_cntrl I/Os 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
10	0x1	CAM: rail cam (called pemmc internally) I/Os 0 = ENABLE 1 = DISABLE
9	0x1	RESERV: Reserved to keep compatibility with Tegra 2 devices and no_iopwr 0 = ENABLE 1 = DISABLE
8	0x1	SD: rail sd I/Os 0 = ENABLE 1 = DISABLE
7	0x1	MEM: rail mem I/Os 0 = ENABLE 1 = DISABLE
6	0x1	LCD: rail lcd I/Os 0 = ENABLE 1 = DISABLE
5	0x1	AUDIO: rail i2s I/Os 0 = ENABLE 1 = DISABLE
4	0x1	VI: rail dvi 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
1	0x1	NAND: rail GMI I/Os (keeping old name for backward compatibility) 0 = ENABLE 1 = DISABLE
0	0x1	SYS: rail AO I/Os 0 = ENABLE 1 = DISABLE

9.6.20 APBDEV_PMC_PWR_DET_LATCH_0

Latches power detect for power rails enabled by Power Detect register.

Power Detect Latch

Offset: 04ch | Read/Write: R/W | Reset: 0b1

Bit	Reset	Description
0	0x1	LATCH: power detect latch, latches value from the pads as long set to 1 0 = ENABLE 1 = DISABLE

9.6.21 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: 050h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SCRATCH0: General-purpose register storage

9.6.22 APBDEV_PMC_SCRATCH1_0

Scratch Register

Offset: 054h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH1: General-purpose register storage

9.6.23 APBDEV_PMC_SCRATCH2_0

Scratch Register

Offset: 058h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH2: General-purpose register storage

9.6.24 APBDEV_PMC_SCRATCH3_0

Scratch Register

Offset: 05ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH3: General-purpose register storage

9.6.25 APBDEV_PMC_SCRATCH4_0

Scratch Register

Offset: 060h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH4: General-purpose register storage

9.6.26 APBDEV_PMC_SCRATCH5_0

Scratch Register

Offset: 064h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH5: General purpose register storage

9.6.27 APBDEV_PMC_SCRATCH6_0

Scratch Register

Offset: 068h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH6: General purpose register storage

9.6.28 APBDEV_PMC_SCRATCH7_0

Scratch Register

Offset: 06ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH7: General purpose register storage

9.6.29 APBDEV_PMC_SCRATCH8_0

Scratch Register

Offset: 070h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH8: General purpose register storage

9.6.30 APBDEV_PMC_SCRATCH9_0

Scratch Register

Offset: 074h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH9: General purpose register storage

9.6.31 APBDEV_PMC_SCRATCH10_0

Scratch Register

Offset: 078h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH10: General purpose register storage

9.6.32 APBDEV_PMC_SCRATCH11_0

Scratch Register

Offset: 07ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH11: General purpose register storage

9.6.33 APBDEV_PMC_SCRATCH12_0

Scratch Register

Offset: 080h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH12: General purpose register storage

9.6.34 APBDEV_PMC_SCRATCH13_0

Scratch Register

Offset: 084h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH13: General purpose register storage

9.6.35 APBDEV_PMC_SCRATCH14_0

Scratch Register

Offset: 088h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH14: General purpose register storage

9.6.36 APBDEV_PMC_SCRATCH15_0

Scratch Register

Offset: 08ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH15: General purpose register storage

9.6.37 APBDEV_PMC_SCRATCH16_0

Scratch Register

Offset: 090h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH16: General purpose register storage

9.6.38 APBDEV_PMC_SCRATCH17_0

Scratch Register

Offset: 094h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH17: General purpose register storage

9.6.39 APBDEV_PMC_SCRATCH18_0

Scratch Register

Offset: 098h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH18: General purpose register storage

9.6.40 APBDEV_PMC_SCRATCH19_0

Scratch Register

Offset: 09ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH19: General purpose register storage

9.6.41 APBDEV_PMC_SCRATCH20_0

Scratch Register

Offset: 0a0h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH20: General purpose register storage

9.6.42 APBDEV_PMC_SCRATCH21_0

Scratch Register

Offset: 0a4h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH21: General purpose register storage

9.6.43 APBDEV_PMC_SCRATCH22_0

Scratch Register

Offset: 0a8h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH22: General purpose register storage

9.6.44 APBDEV_PMC_SCRATCH23_0

Scratch Register

Offset: 0ach | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH23: General purpose register storage

9.6.45 APBDEV_PMC_SECURE_SCRATCH0_0

Secure Scratch Register

Offset: 0b0h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SECURE_SCRATCH0

9.6.46 APBDEV_PMC_SECURE_SCRATCH1_0

Secure Scratch Register

Offset: 0b4h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SECURE_SCRATCH1

9.6.47 APBDEV_PMC_SECURE_SCRATCH2_0

Secure Scratch Register

Offset: 0b8h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SECURE_SCRATCH2

9.6.48 APBDEV_PMC_SECURE_SCRATCH3_0

Secure Scratch Register

Offset: 0bch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SECURE_SCRATCH3

9.6.49 APBDEV_PMC_SECURE_SCRATCH4_0

Secure Scratch Register

Offset: 0c0h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SECURE_SCRATCH4

9.6.50 APBDEV_PMC_SECURE_SCRATCH5_0

Secure Scratch Register

Offset: 0c4h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SECURE_SCRATCH5

9.6.51 APBDEV_PMC_CPUPWRGOOD_TIMER_0

CPU Power Good Timer

SW 33V override - will work only if SW_E33V_OVR_EN is enabled. 1 will set value to 33V, 0 to 2.5

WARNING! Setting 0 on 3.3V interface will damage the chip!

Each rail has an enable bit, only when enable is set, the value is written.

The behavior for timer DATA of zero and one will be the same (i.e., 1 cycle of delay)

Offset: 0c8h | Read/Write: R/W | Reset: 0b00000000000000001111111111111111

Bit	Reset	Description
31:0	0xffff	DATA: timer data

9.6.52 APBDEV_PMC_CPUPWROFF_TIMER_0

CPU Power Off Timer

Used for On to Off transition.

Offset: 0cch | Read/Write: R/W | Reset: 0b00000000000000001111111111111111

Bit	Reset	Description
31:0	0xffff	DATA: timer data

9.6.53 APBDEV_PMC_PG_MASK_0

Offset: 0d0h | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:24	0xff	PX: Mask PCX rail
23:16	0xff	VD: Mask VDE rail
15:8	0xff	VE: Mask VE rail
7:0	0xff	TD: Mask TD rail

9.6.54 APBDEV_PMC_PG_MASK_1_0

Offset: 0d4h | Read/Write: R/W | Reset: 0b11111111111111111111111xxxxxxx1

Bit	Reset	Description
31:24	0xff	SAX: MASK SAX rail
23:16	0xff	HEG: MASK HEG rail
15:8	0xff	MPE: MASK MPE rail
0	0x1	L2C: MASK L2C rail

9.6.55 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: 0d8h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	SMPL: Causes PMC to sample the wake pads 0 = DISABLE 1 = ENABLE

9.6.56 APBDEV_PMC_AUTO_WAKE_LVL_MASK_0

This register is used by s/w to enable sampling of the wake pads before entering LP0

Setting a '1' would cause the associated pad to be sampled and value transferred to WAKE_LVL register

Offset: 0dch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	VALUE

9.6.57 APBDEV_PMC_WAKE_DELAY_0

Offset: 0e0h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	VALUE

9.6.58 APBDEV_PMC_PWR_DET_VAL_0

This register is used to override the power-detect cells and manually set the values (in 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: 0e4h | Read/Write: R/W | Reset: 0b11111111111111

Bit	Reset	Description
13	0x1	SDMMC3: rail sdmmc3 I/Os 0 = ENABLE 1 = DISABLE
12	0x1	SDMMC1: rail sdmmc1 I/Os 0 = ENABLE 1 = DISABLE
11	0x1	PEX_CNTRL: rail pex_cntrl I/Os 0 = ENABLE 1 = DISABLE
10	0x1	CAM: rail cam (called pemmc internally) I/Os 0 = ENABLE 1 = DISABLE
9	0x1	RESERV: reserved to match no_io_pwrrails 0 = ENABLE 1 = DISABLE
8	0x1	SD: rail sd I/Os 0 = ENABLE 1 = DISABLE
7	0x1	MEM: rail mem I/Os 0 = ENABLE 1 = DISABLE
6	0x1	LCD: rail lcd I/Os 0 = ENABLE 1 = DISABLE
5	0x1	AUDIO: rail i2s I/Os 0 = ENABLE 1 = DISABLE
4	0x1	VI: rail dvi 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
1	0x1	NAND: rail at3 I/Os 0 = ENABLE 1 = DISABLE
0	0x1	SYS: rail ao I/Os 0 = ENABLE 1 = DISABLE

9.6.59 APBDEV_PMC_DDR_PWR_0

This register is used to program the "E_18V" pin of the DDR pads.

Offset: 0e8h | Read/Write: R/W | Reset: 0b11

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

9.6.60 APBDEV_PMC_USB_DEBOUNCE_DEL_0

Determines number of 32kHz clock cycles to debounce USB signal events. A value of 0 means only an NP synchronizer to the 32KHz domain, with no debouncing.

Reset values are for Cold Hardware Reset only

Offset: 0ech | Read/Write: R/W | Reset: 0b000000000000000000000000

Bit	Reset	Description
23:20	0x0	UHSIC_LINE_DEB_CNT: Number of 32kHz debounce cycles for UHSIC port 0
19:16	0x0	UTMIP_LINE_DEB_CNT: Number of 32kHz debounce cycles for UTMIP port 0
15:0	0x0	VAL: Debounce period for ID and VBUS events on all USB ports

9.6.61 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 wakeup events.

- USBOP_VAL_PD: Power down for D+ value detector (io pad)
- USBON_VAL_PD: Power down for D- value detector (io 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 wakeup 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: 0f0h | Read/Write: R/W | Reset: 0b1111111111111111

Bit	Reset	Description
15:14	0x3	UHSIC_RESERVED_P0: 2 bits reserved for UHSIC P0
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

Bit	Reset	Description
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

9.6.62 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: 0f4h | Read/Write: R/W | Reset: 0b1

Bit	Reset	Description
0	0x1	VAL: Disabled by default 0 = ENABLE 1 = DISABLE

9.6.63 APBDEV_PMC_PLLP_WB0_OVERRIDE_0

Master control for all wb0 pll override.

Offset: 0f8h | Read/Write: R/W | Reset: 0b0000000000000

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 freq.
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 freq
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: osc 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

9.6.64 APBDEV_PMC_SCRATCH24_0

Scratch Register

Offset: 0fch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH24: General purpose register storage

9.6.65 APBDEV_PMC_SCRATCH25_0

Scratch Register

Offset: 100h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH25: General purpose register storage

9.6.66 APBDEV_PMC_SCRATCH26_0

Scratch Register

Offset: 104h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH26: General purpose register storage

9.6.67 APBDEV_PMC_SCRATCH27_0

Scratch Register

Offset: 108h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH27: General purpose register storage

9.6.68 APBDEV_PMC_SCRATCH28_0

Scratch Register

Offset: 10ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH28: General purpose register storage

9.6.69 APBDEV_PMC_SCRATCH29_0

Scratch Register

Offset: 110h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH29: General purpose register storage

9.6.70 APBDEV_PMC_SCRATCH30_0

Scratch Register

Offset: 114h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH30: General purpose register storage

9.6.71 APBDEV_PMC_SCRATCH31_0

Scratch Register

Offset: 118h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH31: General purpose register storage

9.6.72 APBDEV_PMC_SCRATCH32_0

Scratch Register

Offset: 11ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH32: General purpose register storage

9.6.73 APBDEV_PMC_SCRATCH33_0

Scratch Register

Offset: 120h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH33: General purpose register storage

9.6.74 APBDEV_PMC_SCRATCH34_0

Scratch Register

Offset: 124h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH34: General purpose register storage

9.6.75 APBDEV_PMC_SCRATCH35_0

Scratch Register

Offset: 128h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH35: General purpose register storage

9.6.76 APBDEV_PMC_SCRATCH36_0

Scratch Register

Offset: 12ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH36: General purpose register storage

9.6.77 APBDEV_PMC_SCRATCH37_0

Scratch Register

Offset: 130h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH37: General purpose register storage

9.6.78 APBDEV_PMC_SCRATCH38_0

Scratch Register

Offset: 134h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH38: General purpose register storage

9.6.79 APBDEV_PMC_SCRATCH39_0

Scratch Register

Offset: 138h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH39: General purpose register storage

9.6.80 APBDEV_PMC_SCRATCH40_0

Scratch Register

Offset: 13ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH40: General purpose register storage

9.6.81 APBDEV_PMC_SCRATCH41_0

Scratch Register

Offset: 140h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH41: General purpose register storage

9.6.82 APBDEV_PMC_SCRATCH42_0

Scratch Register

Offset: 144h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH42: General purpose register storage

9.6.83 APBDEV_PMC_BONDOUT_MIRROR0_0

Secure Scratch Register

Offset: 148h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR0

9.6.84 APBDEV_PMC_BONDOUT_MIRROR1_0

Secure Scratch Register

Offset: 14ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR1

9.6.85 APBDEV_PMC_BONDOUT_MIRROR2_0

Secure Scratch Register

Offset: 150h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR2

9.6.86 APBDEV_PMC_SYS_33V_EN_0

Offset: 154h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	VAL: 1 = 3.3v 0 = 1.8v

9.6.87 APBDEV_PMC_BONDOUT_MIRROR_ACCESS_0

On separate reset (same as car) disables access (read/write to secure scratch registers)

Offset: 158h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	BREAD: disable read from bondout secure registers 0 = OFF 1 = ON

Bit	Reset	Description
0	0x0	BWRITE: disable write to bondout secure registers 0 = OFF 1 = ON

9.6.88 APBDEV_PMC_GATE_0

This register is used for SW controlled synchronization between APB domain and the 32 KHz domain. SW 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: 15ch | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	GATE_WAKE: 0 = OFF 1 = ON
0	0x0	GATE_DBNS: 0 = OFF 1 = ON

9.6.89 APBDEV_PMC_WAKE2_MASK_0

New wake registers due to wake events number exceeding 32

wake2 alignments foo masks, level, status

- WAKE2_*[0] = ulpi_data3
- WAKE2_*[1] = gmi_cs0_n
- WAKE2_*[2] = gmi_cs4_n
- WAKE2_*[3] = gmi_cs7_n
- WAKE2_*[4] = sdmmc4_dat1
- WAKE2_*[5] = usb_vbus_wakeup[2]
- WAKE2_*[6] = usb_iddig[2]
- WAKE2_*[7] = utmip0_line_wakeup_event // line wake-up events only status
- WAKE2_*[8] = utmip1_line_wakeup_event // line wake-up events only status
- WAKE2_*[9] = utmip2_line_wakeup_event // line wake-up events only status
- WAKE2_*[10] = uhsic_line_wakeup_event // line wake-up events only status

Offset: 160h | Read/Write: R/W | Reset: 0b00000000000

Bit	Reset	Description
10:7	0x0	EVENT_REST: 0 = DISABLE 1 = ENABLE
6:5	0x0	USB_EVENT: USB 2 events 0 = ACTIVE_LOW 1 = ACTIVE_HIGH
4:0	0x0	EVENT: pin 0-2 wake enable 0 = DISABLE 1 = ENABLE

9.6.90 APBDEV_PMC_WAKE2_LVL_0

This register sets level which is active level for wake event will cause exit from lp0 state if input signal level matches level set in this register and WAKE2_MASK is set for the event to 1 do not need level for 4 line wakeup events, level is always 1.

PMC Wake Level

Offset: 164h | Read/Write: R/W | Reset: 0b01100111

Bit	Reset	Description
7	0x0	ALLOW_PULSE_WAKE: allows pulse wakes in case USB wants to wake on pulse. otherwise it is level wake (signal must last longer than 1ms)
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

9.6.91 APBDEV_PMC_WAKE2_STATUS_0

This register stores status of the wake events. Event will be set if level matches and is not masked

Write will reset wake event set.

PMC wake Status

Offset: 168h | Read/Write: R/W | Reset: 0b00000000000

Bit	Reset	Description
10:7	0x0	EVENT_REST: 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

9.6.92 APBDEV_PMC_SW_WAKE2_STATUS_0

This register stores status of the wake events. Latching of the events in software wake status is enabled by PMC_CNRL register bit LATCHWAKE_EN. Latching will stop at 1 to 0 transition on this bit.

Event will be set if level matches. Masking is not affecting this register. Write will reset wake events set.

PMC software wake status

Offset: 16ch | Read/Write: R/W | Reset: 0b00000000000

Bit	Reset	Description
10:7	0x0	EVENT_REST
6:5	0x0	USB_EVENT: USB wake events 0 = NOT_SET 1 = SET
4:0	0x0	EVENT: pin 0-2 wake

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE

9.6.93 APBDEV_PMC_AUTO_WAKE2_LVL_MASK_0

Offset: 170h | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6:0	0x0	VALUE

9.6.94 APBDEV_PMC_PG_MASK_2_0

Extra PG mask registers added due to new powergated fields. Need 32 bit for CPU now

Offset: 174h | Read/Write: R/W | Reset: 0b1111111111111111

Bit	Reset	Description
15:8	0xff	A9LP: Mask A9LP rail
7:0	0xff	TD2: Mask TD2 rail

9.6.95 APBDEV_PMC_PG_MASK_CE1_0

Offset: 178h | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:0	-1	MASK: Mask CE1 rail

9.6.96 APBDEV_PMC_PG_MASK_CE2_0

Offset: 17ch | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:0	-1	MASK: Mask CE1 rail

9.6.97 APBDEV_PMC_PG_MASK_CE3_0

Offset: 180h | Read/Write: R/W | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31:0	-1	MASK: Mask CE1 rail

9.6.98 APBDEV_PMC_PWRGATE_TIMER_CE_0_0

Separate power_gate_off, on, for CE counters - 6 bit each

Offset: 184h | Read/Write: R/W | Reset: 0b00100100011100010100001100001

Bit	Reset	Description
29:24	0x9	RAIL4: timer value for rail 4
23:18	0x7	RAIL3: timer value for rail 3
17:12	0x5	RAIL2: timer value for rail 2
11:6	0x3	RAIL1: timer value for rail 1

Bit	Reset	Description
5:0	0x1	RAIL0: timer value for rail 0

9.6.99 APBDEV_PMC_PWRGATE_TIMER_CE_1_0

Offset: 188h | Read/Write: R/W | Reset: 0b01001001000000111000110000 1010

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

9.6.100 APBDEV_PMC_PWRGATE_TIMER_CE_2_0

Offset: 18ch | Read/Write: R/W | Reset: 0b01110001101001100001011001 0100

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

9.6.101 APBDEV_PMC_PWRGATE_TIMER_CE_3_0

Offset: 190h | Read/Write: R/W | Reset: 0b10011010010010001010000000 11110

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

9.6.102 APBDEV_PMC_PWRGATE_TIMER_CE_4_0

Offset: 194h | Read/Write: R/W | Reset: 0b11000010111010110010101010 1000

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

9.6.103 APBDEV_PMC_PWRGATE_TIMER_CE_5_0

Offset: 198h | Read/Write: R/W | Reset: 0b11101011100011011010110100110010

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

9.6.104 APBDEV_PMC_PWRGATE_TIMER_CE_6_0

Offset: 19ch | Read/Write: R/W | Reset: 0b111100111011

Bit	Reset	Description
11:6	0x3c	RAIL31: timer value for rail 31
5:0	0x3b	RAIL30: timer value for rail 30

9.6.105 APBDEV_PMC_PCX_EDPD_CNTRL_0

Extra secure scratch register

Offset: 1a0h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	EN: when set sets the edpd to pcx (pll toggle) 0 = OFF 1 = ON

9.6.106 APBDEV_PMC_OSC_EDPD_OVER_0

This register can be programmed to keep oscillator ON during lp0 mode. It cuts the latency time on lp0 wake if oscillator pad does not have to be restarted.

When oscillator pad is on, during lp0 mode, dspare, duty, strength, bypass are controlled by fields below.

Oscillator can be in one of four modes during lp0 - in dpd, not dpd but not enabled, running, running only when external request is active.

Offset: 1a4h | Read/Write: R/W | Reset: 0b00000000000000001111110

Bit	Reset	Description
22	0x0	OSC_CTRL_SELECT: select whether CAR's OSC_CTRL or 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 oscillator in dpd mode during LP0, bypass dpd, but do not enable oscillator during LP0, force the oscillator to run during LP0, run the oscillator when requested by an external clock request pin 0 = DPD 1 = OFF 2 = ON 3 = EXT_REQ

Bit	Reset	Description
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).

9.6.107 APBDEV_PMC_CLK_OUT_CNTRL_0

This register was added to control 3 clock outputs of the chip. Each of the clock outputs can be in one of 4 modes - not running, running when request is active high, running when request is active low, always running clock output when not running can be tristated, high or low.

Clock source might be from car unit (not avail when in lp0 mode), Osc, osc_div2, osc_div4

Offset: 1a8h | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
23:22	0x0	CLK3_SRC_SEL: 0 = OSC_DIV1 1 = OSC_DIV2 2 = OSC_DIV3 3 = CAR
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

Bit	Reset	Description
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:

9.6.108 APBDEV_PMC_SATA_PWRGT_0

Added to control sata plls

PLLE or padpll can be disabled/enabled by sw or hw request.

Offset: 1ach | Read/Write: R/W | Reset: 0b00111111

Bit	Reset	Description
7	0x0	SW_PLL_EDPD: sata pll put in dpd state when bit is set, independently of powergating - 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 Sw can put the plle in iddq 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 HW (SATA +AFI+CAR) signals. 1 The PLLE is put in IDDQ by SW -- default 0 = OFF 1 = ON
3	0x1	PADPHY_IDDQ_OVERRIDE_VALUE: 0 The phy is powered up 1 SW can put the PHY in iddq 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 HW (SATA +AFI+CAR) signals. 1 The SATA PHY is put in IDDQ by SW - default 0 = OFF 1 = ON
1	0x1	PADPLL_IDDQ_OVERRIDE_VALUE: 0 The pad PLL is powered up 1 SW can put the pad PLL in IDDQ 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 HW (SATA +AFI+CAR) signals.1 The SATA pad PLL is put in IDDQ by SW. - default 0 = OFF 1 = ON

9.6.109 APBDEV_PMC_SENSOR_CTRL_0

For sensor shutdown and control.

Sensor control register defines chip behavior when sensor request is triggered. It can power gate down all CPUs if enabled. It can trigger reset - (will cause CPU power request to go down and power gating down all CPUs)

IMPORTANT-- to preserve RAM content, any reads/writes to scratch registers will be blocked after sensor triggered reset.

SW must reset BLOCK_SCRATCH_WRITE to enable writes to scratch registers.

Offset: 1b0h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2	0x0	BLOCK_SCRATCH_WRITE: reset by user, set by 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 gate cpus on temp sensor going up 0 = OFF 1 = ON

9.6.110 APBDEV_PMC_RST_STATUS_0

Simplified reset and reset source

Offset: 1b4h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2:0	0x0	RST_SOURCE: source of reset 0 = POR 1 = WATCHDOG 2 = SENSOR 3 = SW_MAIN 4 = LP0

9.6.111 APBDEV_PMC_IO_DPD_REQ_0

Puts I/O rails in DPD mode 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. Register will be still updated.

Setting IO_DPD_REQ should be preceded by writing to DPD_SAMPLE registers to enable snapshot of data to be driven and SEL_DPD_TIM register. SEL_DPD_TIM is set to safe value, but it can be lowered if sys clock is slower. (minimum 200 ns required)

DPD Request

Offset: 1b8h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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: 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 xm2_data0_pad, xm2_data1_pad, xm2_data2_pad, xm2_data3_pad, xm2_mckfb0_pad, xm2_mckfb1_pad, xm2_mckfb2_pad, xm2_mckfb3_pad in/out of deep power down mode 0 = OFF 1 = ON

Bit	Reset	Description
26	0x0	DISC_ADDR_CMD: puts xm2_mclk_pad, xm2_mcke0_pad, xm2_mcke1_pad, xm2_mcsc_n_pad, xm2_mcsd_n_pad, xm2_ma[9:0] in/out of deep power down mode 0 = OFF 1 = ON
25	0x0	DDR_ADDR_CMD: puts 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 in/out of deep power down mode 0 = OFF 1 = ON
24	0x0	POP_ADDR_CMD: puts 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 in/out of deep power down mode 0 = OFF 1 = ON
23	0x0	POP_CLK: puts xm2_mclk_addr_pad in/out of deep power down mode 0 = OFF 1 = ON
22	0x0	COMP: puts xm2_comp_pd_pad, xm2_comp_pu_pad in/out of deep power down mode
21	0x0	DISC_VTTGEN: puts 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 in/out of deep power down mode 0 = OFF 1 = ON
20	0x0	POP_VTTGEN: put pvi_vttgen_1_pad in/out of deep power down mode 0 = OFF 1 = ON
19	0x0	HSIC: puts HSIC rail in/out of deep power down mode 0 = OFF 1 = ON
18	0x0	LCD: puts LCD 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: puts vi rail in/out of deep power down mode 0 = OFF 1 = ON
15	0x0	BB: puts BB rail in/out of deep power down mode 0 = OFF 1 = ON
14	0x0	UART: puts UART rail in/out of deep power down mode 0 = OFF 1 = ON
13	0x0	NAND: puts 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
10	0x0	USB1: puts USB1 in/out of deep power down mode 0 = OFF 1 = ON

Bit	Reset	Description
9	0x0	USB0: puts USB0 in/out of deep power down mode 0 = OFF 1 = ON
8	0x0	DAC: puts dac in/out of deep power down mode 0 = OFF 1 = ON
7	0x0	PEX_CLK3: puts pex rail 2 in/out of deep power down mode 0 = OFF 1 = ON
6	0x0	PEX_CLK2: puts pex rail 1 in/out of deep power down mode 0 = OFF 1 = ON
5	0x0	PEX_CLK1: puts pex rail 1 in/out of deep power down mode 0 = OFF 1 = ON
4	0x0	PEX_BIAS: puts pex rail 0 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

9.6.112 APBDEV_PMC_IO_DPD_STATUS_0

Reflects DPD status of each I/O rail.

DPD status

Offset: 1bch | Read/Write: R/W | Reset: 0b00000000000000000000000000xxxx

Bit	R/W	Reset	Description
29	RW	0x0	USB_IC: 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
27	RW	0x0	DDR_DATA: xm2_data0_pad, xm2_data1_pad, xm2_data2_pad, xm2_data3_pad, xm2_mckfb0_pad, xm2_mckfb1_pad, xm2_mckfb2_pad, xm2_mckfb3_pad in deep power down mode 0 = OFF 1 = ON
26	RW	0x0	DISC_ADDR_CMD: xm2_mclk_pad, xm2_mcke0_pad, xm2_mcke1_pad, xm2_mcsc_n_pad, xm2_mcsd_n_pad, xm2_ma[9:0] in deep power down mode 0 = OFF 1 = ON

Bit	R/W	Reset	Description
25	RW	0x0	DDR_ADDR_CMD: 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 in deep power down mode 0 = OFF 1 = ON
24	RW	0x0	POP_ADDR_CMD: 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 in deep power down mode 0 = OFF 1 = ON
23	RW	0x0	POP_CLK: xm2_mclk_addr_pad in deep power down mode 0 = OFF 1 = ON
22	RW	0x0	COMP: xm2_comp_pd_pad, xm2_comp_pu_pad in deep power down mode
21	RW	0x0	DISC_VTTGEN: 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 in deep power down mode 0 = OFF 1 = ON
20	RW	0x0	POP_VTTGEN: put pvi_vttgen_1_pad in deep power down mode 0 = OFF 1 = ON
19	RW	0x0	HSIC: hsic rail in deep power down mode 0 = OFF 1 = ON
18	RW	0x0	LCD: lcd rail in deep power down mode 0 = OFF 1 = ON
17	RW	0x0	AUDIO: audio rail in deep power down mode 0 = OFF 1 = ON
16	RW	0x0	VI: 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
10	RW	0x0	USB1: usb1 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
8	RW	0x0	DAC: dac in deep power down mode 0 = OFF 1 = ON
7	RW	0x0	PEX_CLK3: pex rail 2 in deep power down mode 0 = OFF 1 = ON
6	RW	0x0	PEX_CLK2: pex rail 1 in deep power down mode 0 = OFF 1 = ON
5	RW	0x0	PEX_CLK1: pex rail 1 in deep power down mode 0 = OFF 1 = ON
4	RW	0x0	PEX_BIAS: pex rail 0 in deep power down mode 0 = OFF 1 = ON
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

9.6.113 APBDEV_PMC_IO_DPD2_REQ_0

DPD request needs second set due to additional rails.
Offset: 1c0h | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6:5	0x0	CODE: code of operation for all set bits 0 = IDLE 1 = DPD_OFF 2 = DPD_ON
4	0x0	CAM: puts cam 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: puts pex_cntrl in/out of deep power down mode 0 = OFF 1 = ON

9.6.114 APBDEV_PMC_IO_DPD2_STATUS_0

DPD2 status

Offset: 1c4h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	CAM: cam in/out of deep power down mode 0 = OFF 1 = ON
3	0x0	SDMMC4: SDMMC4 in/out of deep power down mode 0 = OFF 1 = ON
2	0x0	SDMMC3: SDMMC3 in/out of deep power down mode 0 = OFF 1 = ON
1	0x0	SDMMC1: SDMMC1 in/out of deep power down mode 0 = OFF 1 = ON
0	0x0	PEX_CNTRL: pex_cntrl in/out of deep power down mode 0 = OFF 1 = ON

9.6.115 APBDEV_PMC_SEL_DPD_TIM_0

This timer is added to guarantee proper timing spacing in HW between sel_dpd and e_dpd signals issued to pads.

Minimum of 200ns is required, timer in APB/sys clock.

Offset: 1c8h | Read/Write: R/W | Reset: 0b1111111

Bit	Reset	Description
6:0	0x7f	SEL_DPD_TIM: timer which separates e_dpd deassertion time from sel_dpd deassertion time. in apb_clk

9.6.116 APBDEV_PMC_VDDP_SEL_0

Power set for new DDR pads. Safe value is 11.

Offset: 1cch | Read/Write: R/W | Reset: 0b11

Bit	Reset	Description
1:0	0x3	DATA: vddp sel bits to DDR pads

9.6.117 APBDEV_PMC_DDR_CFG_0

Package type for CAR/PMC Control

Offset: 1d0h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	IF: 0 = LPDDR2 1 = DDR3
0	0x0	PKG: package type 0 = DISC 1 = POP

9.6.118 APBDEV_PMC_E_NO_VTTGEN_0

Offset: 1d4h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2:0	0x0	MASK: when set sets E_NO_VTTGEN to 1 on rail during DPD mode only, (turns of DDR3 VTTGEN cell)

9.6.119 APBDEV_PMC_PLLM_WB0_OVERRIDE_FREQ_0

New PLL WB override registers to accelerate warm boot time

Offset: 1dch | Read/Write: R/W | Reset: 0b00001000000000000000000000101100

Bit	Reset	Description
30	0x0	PLLM_DCCON: Base PLLM DCCON control.
29:26	0x1	PLLM_CPCON: Base PLLM charge pump setup control.
25:22	0x0	PLLM_LFCON: Base PLLM loop filter setup control.
21:18	0x0	PLLM_VCOCON: Base PLLM VCO range setup control.
17:15	0x0	PLLM_DIVP: 0 = post divider (2^n).
14:5	0x1	PLLM_DIVN: PLL feedback divider.
4:0	0xc	PLLM_DIVM: PLL input divider.

9.6.120 APBDEV_PMC_TEST_PWRGATE_0

Force test power gate override off/on

Offset: 1e0h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
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

9.6.121 APBDEV_PMC_PWRGATE_TIMER_MULT_0

Adding register which would allow for multiply of power gating time set by PWRGATE_TIMER_OFF.

Need separate multiplier for CPU, since timing requirements might be different.

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 CPU power-gated case, MULT_CPU will be used with PWRGATE_TIMER_CE* registers value. All timers are in sys_clk units.

Offset: 1e4h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3:2	0x0	MULT_CPU: 0 = ONE 1 = TWO 2 = FOUR
1:0	0x0	MULT: 0 = ONE 1 = TWO

Bit	Reset	Description
		2 = FOUR

9.6.122 APBDEV_PMC_DSI_SEL_DPD_0

Register to control sel_dpd for DSI pad. Allows driving lp0 value on DSI pad beyond lp0 exit. This enables DSI to be programmed properly at lp0 exit while lp0 value is still driven by brick pad.

Offset: 1e8h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	SET_DSIB: 0 = OFF 1 = ON
0	0x0	SET_DSIA: 0 = OFF 1 = ON

9.6.123 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. 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: 1ech | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

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 slepp 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

9.6.124 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: 1f0h | Read/Write: R/W | Reset: 0b00001111000011110000111100001111

Bit	Reset	Description
31	0x0	UHSIC_WAKE_EX_P0: wakeup 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: wakeup 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: wakeup 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: wakeup 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

9.6.125 APBDEV_PMC_UTMIP_PAD_CFG_0

Set of static config values for USB I/O pads under DPD mode. These were config 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: 1f4h | Read/Write: RO | Reset: 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

9.6.126 APBDEV_PMC_UTMIP_TERM_PAD_CFG_0

Set of termination config values for USB I/O pads under DPD mode. These are config values 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 config for all UTMIP ports

Offset: 1f8h | Read/Write: R/W | Reset: 0b0100001000

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

9.6.127 APBDEV_PMC_UTMIP_UHSIC_SLEEP_CFG_0

Register that config the value of the line that could cause a wakeup 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: 1fch | Read/Write: R/W | Reset: 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

9.6.128 APBDEV_PMC_UTMIP_UHSIC_SLEEPWALK_CFG_0

Register determines when sleep walking should take effect. Upon a specific GPIO event, on any wake event or on a linevalue 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: 200h | Read/Write: R/W | Reset: 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

9.6.129 APBDEV_PMC_UTMIP_SLEEPWALK_P0_0

Registers that 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 pin MASTER_ENABLE 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 wakeup events.

Signaling Sequence for UTMIP Port 0 Wakeup

Offset: 204h | Read/Write: R/W | Reset: 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

Bit	Reset	Description
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
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

9.6.130 APBDEV_PMC_UTMIP_SLEEPWALK_P1_0

Signaling Sequence for UTMIP Port 1 Wakeup

Offset: 208h | Read/Write: R/W | Reset: 0b00100011001000011001000 01101100011

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

Bit	Reset	Description
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
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

9.6.131 APBDEV_PMC_UTMIP_SLEEPWALK_P2_0

Signaling Sequence for UTMIP Port 2 Wakeup

Offset: 20ch | Read/Write: R/W | Reset: 0b00100011001000011001000110110011

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

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, 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

9.6.132 APBDEV_PMC_UHSIC_SLEEPWALK_P0_0

Registers that 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 dangling (unconnected) until a time that departure may be implemented.

Signaling Sequence for UHSIC Port 0 Wakeup

Offset: 210h | Read/Write: R/W | Reset: 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

Bit	Reset	Description
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

9.6.133 APBDEV_PMC_UTMIP_UHSIC_STATUS_0

Read only register that provides current walk pointer information as well as line value.

Status of UTMIP UHSIC wakeup circuitry

Offset: 214h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxx

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

9.6.134 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: 218h | Read/Write: R/W | Reset: 0b00010001000100010001000100010001

Bit	Reset	Description
27	0x0	UTMIP_ID_FAKE_EN_P2: Enable the fake ID value for UTMIP P2
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

9.6.135 APBDEV_PMC_BONDOUT_MIRROR3_0

Secure scratch register

Offset: 21ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR3

9.6.136 APBDEV_PMC_BONDOUT_MIRROR4_0

Secure scratch register

Offset: 220h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	BONDOUT_MIRROR4

9.6.137 APBDEV_PMC_SECURE_SCRATCH6_0

Secure scratch register

Offset: 224h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SECURE_SCRATCH6

9.6.138 APBDEV_PMC_SECURE_SCRATCH7_0

Secure scratch register

Offset: 228h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SECURE_SCRATCH7

9.6.139 APBDEV_PMC_SCRATCH43_0

Scratch register

Offset: 22ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH43: General purpose register storage

9.6.140 APBDEV_PMC_SCRATCH44_0

Scratch register

Offset: 230h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH44: General purpose register storage

9.6.141 APBDEV_PMC_SCRATCH45_0

Scratch register

Offset: 234h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH45: General purpose register storage

9.6.142 APBDEV_PMC_SCRATCH46_0

Scratch register

Offset: 238h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH46: General purpose register storage

9.6.143 APBDEV_PMC_SCRATCH47_0

Scratch register

Offset: 23ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH47: General purpose register storage

9.6.144 APBDEV_PMC_SCRATCH48_0

Scratch register

Offset: 240h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH48: General purpose register storage

9.6.145 APBDEV_PMC_SCRATCH49_0

Scratch register

Offset: 244h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH49: General purpose register storage

9.6.146 APBDEV_PMC_SCRATCH50_0

Scratch register

Offset: 248h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH50: General purpose register storage

9.6.147 APBDEV_PMC_SCRATCH51_0

Scratch register

Offset: 24ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH51: General purpose register storage

9.6.148 APBDEV_PMC_SCRATCH52_0

Scratch register

Offset: 250h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH52: General purpose register storage

9.6.149 APBDEV_PMC_SCRATCH53_0

Scratch register

Offset: 254h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH53: General purpose register storage

9.6.150 APBDEV_PMC_SCRATCH54_0

Scratch register

Offset: 258h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH54: General purpose register storage

9.6.151 APBDEV_PMC_SCRATCH55_0

Scratch register

Offset: 25ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SCRATCH55: General purpose register storage

9.6.152 APBDEV_PMC_SCRATCH0_ECO_0

Scratch register

Offset: 2fch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	ECO0: Reserved

9.6.153 APBDEV_PMC_SCRATCH1_ECO_0

Scratch register

Offset: 300h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	ECO1: Reserved

9.6.154 APBDEV_PMC_SCRATCH2_ECO_0

Scratch register

Offset: 304h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	ECO2: Reserved

9.6.155 APBDEV_PMC_UTMIP_UHSIC_LINE_WAKEUP_0

Offset: 308h | Read/Write: R/W | Reset: 0b1111

Bit	Reset	Description
3	0x1	UHSIC_LINE_WAKEUP_EN_P0: enables latching line wake-up event on UHSIC p0
2	0x1	UTMIP_LINE_WAKEUP_EN_P2: enables latching line wake-up event on UTMIP p2
1	0x1	UTMIP_LINE_WAKEUP_EN_P1: enables latching line wake-up event on UTMIP p1
0	0x1	UTMIP_LINE_WAKEUP_EN_P0: enables latching line wake-up event on UTMIP p0

9.6.156 APBDEV_PMC_UTMIP_BIAS_MASTER_CNTRL_0

Offset: 30ch | Read/Write: R/W | Reset: 0b1101

Bit	Reset	Description
3	0x1	UTMIP_BIAS_MASTER_AWAKE_AND: When not PROGRAMMABLE and E_DPD=0, AND the MASTER_ENABLE of all I/O 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 I/O 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

9.6.157 APBDEV_PMC_UTMIP_MASTER_CONFIG_0

MASTER_CONFIG selects the MASTER function's mode of operation, in this case 0 for 500uA usage, 1 for smaller current usage when driving.

Offset: 310h | Read/Write: R/W | Reset: 0b0000

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

Bit	Reset	Description
1	0x0	UTMIP_PWR_P1: enables UTMIP p1 low power mode
0	0x0	UTMIP_PWR_P0: enables UTMIP p0 low power mode

9.7 Secure Boot Registers

9.7.1 SB_CSR_0

Secure Boot Control Status Register

Offset: 0x000 | Read/Write: R/W | Reset: 0x00000021 (0b000000000010xx01)

Bit	R/W	Reset	Description
15:12	RW	0x0	COT_FAIL_COUNT: Chain-of-Trust Fail Count
11:8	RW	0x0	SWDM_FAIL_COUNT: Secure Watchdog Monitor fail Count
7	RW	0x0	SWDM_ENABLE: Secure Watchdog Monitor Enable 1 = ENABLE 0 = DISABLE
6	RW	0x0	HANG: Secure Boot Hang 1 = ENABLE 0 = DISABLE
5	RW	0x1	JTAG_DISABLE: Secure JTAG Disable 0 = ENABLE 1 = DISABLE
4	RW	0x0	PIROM_DISABLE: Protected iROM Disable 0 = ENABLE 1 = DISABLE
3:2	RO	X	Rsrvd_31: Reserved
1	RW	0x0	NS_RST_VEC_WR_DIS: Non-secure reset vector write disable 0 = ENABLE 1 = DISABLE
0	RW	0x1	SECURE_BOOT_FLAG: 1 = Booted into Secure Mode 1 = ENABLE 0 = DISABLE

9.7.2 SB_PIROM_START_0

This specifies the offset from the start of the Boot ROM to the protected Region. This register is only programmable while in Secure_Mode (SECURE_BOOT_FLAG above == 1)

The lower 7 bits (6:0) are not significant and are assumed to be zero.

Secure Boot Protected ROM Start

Offset: 0x004 | Read/Write: R/W | Reset: 0x00001000 (0b00000000000000000001000000000000)

Bit	Reset	Description
31:0	0x1000	PROTECTED_ROM_START: PROTECTED_ROM_START

9.7.3 SB_PFCFG_0

Secure Boot Processor Feature Configuration Register

Offset: 0x008 | Read/Write: R/W | Reset: 0x00c0008f (0b0110xxxxxxxx0000010001111)

Bit	R/W	Reset	Description
24	RW	0x0	CS_DEADSTATUS_EN: Coresight™ Debug bit to enable return codes on bad accesses 1 = ENABLE 0 = DISABLE
23	RW	0x1	CS_ACCESS_CHECK_EN: Debug bit to Enable Checking for bad Csite accesses such as when CPU not in active partition 1 = ENABLE 0 = DISABLE
22	RW	0x1	CS_CODES_EN: Coresight Debug bit to Enable Status Codes 1 = ENABLE 0 = DISABLE
21	RW	0x0	CS_TIMEOUT_TM: Coresight Timeout Testmode 1 = ENABLE 0 = DISABLE
20	RO	X	DBGACK_AVP: DBGACKnowledge Status from AVP 1 = ENABLE 0 = DISABLE
19	RO	X	DBGACK_CPU3: DBGACKnowledge Status from CPU3 1 = ENABLE 0 = DISABLE
18	RO	X	DBGACK_CPU2: DBGACKnowledge Status from CPU2 1 = ENABLE 0 = DISABLE
17	RO	X	DBGACK_CPU1: DBGACKnowledge Status from CPU1 1 = ENABLE 0 = DISABLE
16	RO	X	DBGACK_CPU0: DBGACKnowledge Status from CPU0. The following bits assert the Processor External Debug Request signal. Setting these bits can cause a break. Clearing these bits will NOT restart the processor, but the request needs to be cleared before the debugger can restart the processor 1 = ENABLE 0 = DISABLE
12	RW	0x0	EDBGRQ_AVP: External Debug Request for AVP 1 = ENABLE 0 = DISABLE
11	RW	0x0	EDBGRQ_CPU3: External Debug Request for CPU3 via CoreSight 1 = ENABLE 0 = DISABLE
10	RW	0x0	EDBGRQ_CPU2: External Debug Request for CPU2 via CoreSight 1 = ENABLE 0 = DISABLE
9	RW	0x0	EDBGRQ_CPU1: External Debug Request for CPU1 via CoreSight 1 = ENABLE 0 = DISABLE
8	RW	0x0	EDBGRQ_CPU0: External Debug Request for CPU0 via CoreSight.Coresight Timeout Enable for bus transactions 1 = ENABLE 0 = DISABLE
7	RW	0x1	CS_TIMEOUT_EN: Coresight Timeout Enable RTCK delay control 1 = ENABLE 0 = DISABLE

Bit	R/W	Reset	Description
6	RW	0x0	CS_RTCK_SPEED: RTCK delay (Synchronized or Direct) The following bits configure the Processor Feature pins. Once these bit values are assigned to the Most-Secure mode, they cannot be flipped back again 0 = SLOW 1 = FAST
5	RW	0x0	CP15SDISABLE: Disable access to system control registers enum (DISABLE=0x1,ENABLE=0x0) 1 = DISABLE 0 = ENABLE
4	RW	0x0	CFGSDISABLE: Disable access to secure control registers 1 = DISABLE 0 = ENABLE
3	RW	0x1	DBGEN: Debug Enable (Not the same as JTAG enable) 1 = ENABLE 0 = DISABLE
2	RW	0x1	NIDEN: Non-Invasive Debug Enable 1 = ENABLE 0 = DISABLE
1	RW	0x1	SPIDEN: Secure Privileged Invasive Debug Enable 1 = ENABLE 0 = DISABLE
0	RW	0x1	SPNIDEN: Secure Privileged Non-Invasive Debug Enable 1 = ENABLE 0 = DISABLE

10.0 REAL-TIME CLOCK

The Real-Time Clock (RTC) module maintains seconds and milliseconds counters, and five alarm registers. RTC is in '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 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.

10.1 Functional Description

The RTC operates in two clock domains; APB clock domain, and 32 KHz clock domain. RTC continues updating the millisecond and second 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 32 KHz clock domain. Writes are transferred to 32 KHz domain with BUSY.STATUS set as BUSY until the transfer is completed. Reads are shadowed in APB clock domain and return immediately.

RTC implements a Seconds Counter register, three Alarm registers, two countdown alarms, and various interrupt related registers.

At boot, the seconds counter is updated to reflect the current time. Writes to seconds counter can be disabled by writing to 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 32K clock domain. Writes are transferred to 32K domain with BUSY STATUS set as BUSY until the transfer completes. Reads are shadowed in APB domain and will return immediately.

Resets

RTC receives an asynchronous reset which is synchronized to APB clock and RTC clock domains.

10.2 RTC Registers

10.2.1 APBDEV_RTC_CONTROL_0

Control register.

Offset: 000h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	WR_SEC_CNT: When set, writes to SECONDS counter are disabled. can only cleared by resetting the RTC module 0 = DISABLE 1 = ENABLE

10.2.2 APBDEV_RTC_BUSY_0

Busy register.

Offset: 004h | Read/Write: RO | Reset: 0bx

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

10.2.3 APBDEV_RTC_SECONDS_0

Seconds counter register.

SECONDS register is copied over to APB side every eight 32KHz clocks (~250uS). 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: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SECONDS: seconds counter is incremented for every 1000 milli-seconds

10.2.4 APBDEV_RTC_SHADOW_SECONDS_0

Shadowed seconds counter register.

Shadow SECONDS register is updated over to 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 is snapshotted in this Register. If the software needs to read two registers, then it should read MILLI_SECONDS Register and then this Register. It should not read the SECONDS register, as it contains the updated value.

Offset: 00ch | Read/Write: RO | Reset: 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.

10.2.5 APBDEV_RTC_MILLI_SECONDS_0

Milliseconds counter register.

Milli SECONDS register is copied over to APB side every eight 32KHz clocks (~250uS). Because of this, performing a read immediately after a write might return old value.

Offset: 010h | Read/Write: RO | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	MILLI_SECONDS: milliseconds counter is incremented using Bresenham algorithm

10.2.6 APBDEV_RTC_SECONDS_ALARM0_0

Seconds alarm0 register.

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: 014h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SECS_MATCH_VALUE: match value to trigger the alarm

10.2.7 APBDEV_RTC_SECONDS_ALARM1_0

Seconds alarm1 register.

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: 018h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SECS_MATCH_VALUE: match value to trigger the alarm

10.2.8 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: 01ch | Read/Write: R/W | Reset: 0b0000000000

Bit	Reset	Description
9:0	0x0	MSEC_MATCH_VALUE: milliseconds match value.

10.2.9 APBDEV_RTC_SECONDS_COUNTDOWN_ALARM_0

Countdown alarm registers.

If ENABLE_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_1 is set after the specified number of milliseconds have elapsed. If REPEAT is ENABLED, the countdown operation is performed repeatedly.

Offset: 020h | Read/Write: R/W | Reset: 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

10.2.10 APBDEV_RTC_MILLI_SECONDS_COUNTDOWN_ALARM_0

Offset: 024h | Read/Write: R/W | Reset: 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

10.2.11 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: 028h | Read/Write: R/W | Reset: 0b00000

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

10.2.12 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: 02ch | Read/Write: R/W | Reset: 0b00000

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

10.2.13 APBDEV_RTC_INTR_SOURCE_0

Interrupt source register.

This is a read-only register which returns the AND of interrupt source and interrupt mask registers.

Offset: 030h | Read/Write: R/W | Reset: 0b00000

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

10.2.14 APBDEV_RTC_INTR_SET_0

Interrupt set register.

This is a write-only register which 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. Read always returns 'h0.

Offset: 034h | Read/Write: R/W | Reset: 0b00000

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

10.2.15 APBDEV_RTC_CORRECTION_FACTOR_0

Correction factor (digital trimming) register

Support is for +/- 500ppm, but +/- 50 ppm is maximum

Offset: 038h | Read/Write: R/W | Reset: 0b0000000000

Bit	Reset	Description
9	0x0	DIRECTION: 0 = DECREMENT 1 = INCREMENT
8:0	0x0	PPM

11.0 GR2D

11.1 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.

11.2 Registers

11.2.1 G2SB_G2SBFORMAT_0

Offset: 0x1c │ Byte Offset: 0x70 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxx)

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

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>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
6 = Y8V8Y8U8_TC
7 = V8Y8U8Y8_TC
8 = B5G6R5</td></tr>
</table>

Bit	Reset	Description
		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

11.2.2 G2SB_G2CONTROLSB_0

Offset: 0x1d │ Byte Offset: 0x74 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000)

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.

Bit	Reset	Description
		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
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 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 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

Bit	Reset	Description
		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 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.

11.2.3 G2SB_G2CONTROLSECOND_0

Offset: 0x1e │ Byte Offset: 0x78 │ Read/Write: R/W │ Reset: 0x00000000 (0b000000x00xxxx000xxxx000000000)

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 clock wise by 90 degrees) 101 = ROT_180 110 = ROT_270 (clock wise 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) Max 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, sw 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

Bit	Reset	Description
		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 - dest base address DSTS - dest 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, line8 to line15 will be processed twice. 00 = disable fast rotate - this turns off the 2nd 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: 32bits 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 1bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero PL2BPP: ALPHA blending: Alpha 2bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero. PL4BPP: ALPHA blending: Alpha 4bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero. PL8BPP: ALPHA blending: Alpha 8bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1bit from source B5G5R5A1, alpha(MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4bits 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 hw by vcaa engine; not really a vcaa resolve) PLS8BPP: ALPHA blending: Alpha 8bits 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 8bits 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 128bit boundary. 2. Destination width has to be multiple of 4 pixels. 3. Source/Destination strides have to be multiple of 128bits. 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 1=enable 0= disable 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 1=enable 0= disable 1= enable 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 1=enable 0= disable 1= enable 0 = DISABLE 1 = ENABLE

11.2.4 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 doesn't 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 (ex: BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
- ALPTYPE_PLS4BPP: xxxA_4444 (ex: BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
- (other): xxx_565 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])

- color depth BPP32:
  - (all) : xxxA_8888 (ex: 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 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (ex: 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();
```

SW *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 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 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 (0b000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: pattern Data Select: SRCSEL and PATSEL can't 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, don't send signal to Display to switch buffer 01 two buffers, dstba and dstba_b are used11 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

Bit	Reset	Description
		lsb) is the first bit If HLMONO is 0, bit 7 (the msb) is the first bit
20	X	SRCCD:0 Source mono1 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 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 tell you where to start in the 16x16 tile when expanding

Bit	Reset	Description

		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 128bit/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 in executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

11.2.5 G2SB_G2ROPFADE_0

Offset: 0x20 │ Byte Offset: 0x80 │ 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.

11.2.6 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

11.2.7 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

11.2.8 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

11.2.9 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 (0bxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

11.2.10 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

11.2.11 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 bytes aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

11.2.12 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

11.2.13 G2SB_G2PATBGC_0

Offset: 0x28 | Byte Offset: 0xa0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

11.2.14 G2SB_G2PATFGC_0

Offset: 0x29 | Byte Offset: 0xa4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

11.2.15 G2SB_G2PATKEY_0

Offset: 0x2a | Byte Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

11.2.16 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)

11.2.17 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

11.2.18 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)

11.2.19 G2SB_G2DSTST_0

Offset: 0x2e | Byte Offset: 0xb8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

11.2.20 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 (0bxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

11.2.21 G2SB_G2SRCPACK_SIZE_0

In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, it is required width/stride be 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

11.2.22 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)

11.2.23 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-bit, all 8 byte-positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there are no restrictions on this value.)

11.2.24 G2SB_G2SRCST_0

Offset: 0x33 | Byte Offset: 0xcc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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.

11.2.25 G2SB_G2SRCBGC_0

srcColors

Offset: 0x34 | Byte Offset: 0xd0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

11.2.26 G2SB_G2SRCFGC_0

Offset: 0x35 | Byte Offset: 0xd4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

11.2.27 G2SB_G2SRCKEY_0

Offset: 0x36 | Byte Offset: 0xd8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

11.2.28 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

11.2.29 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

11.2.30 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

11.2.31 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

11.2.32 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

11.2.33 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 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 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)

11.2.34 G2SB_G2LINESETTING_0

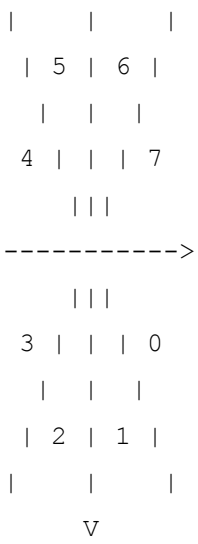
Line Methods

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
25	X	LINEXDIR
24	X	MAJOR:0:xmajor 1: y major

Bit	Reset	Description
		0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

11.2.35 G2SB_G2LINEDELTAN_0



Offset: 0x3e | Byte Offset: 0xf8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

11.2.36 G2SB_G2LINEDELTAM_0

Offset: 0x3f | Byte Offset: 0xfc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

11.2.37 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

11.2.38 G2SB_G2LINELEN_0

Offset: 0x41 | Byte Offset: 0x104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

11.2.39 G2SB_G2CSCFOURTH_0

Offset: 0x42 | Byte Offset: 0x108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.2.40 G2SB_G2SRCST_B_0

Offset: 0x43 | Byte Offset: 0x10c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

11.2.41 G2SB_G2UVSTRIDE_0

Offset: 0x44 | Byte Offset: 0x110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

11.2.42 G2SB_G2CBDES2_0

Circular buffer controller 2

Offset: 0x45 | Byte Offset: 0x114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

11.2.43 G2SB_G2TILEMODE_0

Offset: 0x46 | Byte Offset: 0x118 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxx0xxx0xxx0xxx0xxx0)

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

Bit	Reset	Description
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

11.2.44 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

11.2.45 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

11.2.46 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

11.2.47 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

11.2.48 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

11.2.49 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

11.3 G2SB CTX1 Registers

11.3.1 G2SB_CTX1_INCR_SYNCPT_0

Memory Controller Tiling definitions

Offset: 0x4000 | Byte Offset: 0x10000 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

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

11.3.2 G2SB_CTX1_INCR_SYNCPT_CNTRL_0

Offset: 0x4001 | Byte Offset: 0x10004 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx0)

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.

11.3.3 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.

11.3.4 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 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

11.3.5 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 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

11.3.6 G2SB_CTX1_G2TRIGGER1_0

Offset: 0x400a | Byte Offset: 0x10028 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

11.3.7 G2SB_CTX1_G2TRIGGER2_0

Offset: 0x400b | Byte Offset: 0x1002c | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

11.3.8 G2SB_CTX1_G2CMDSEL_0

Offset: 0x400c | Byte Offset: 0x10030 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxxxxxx00x0x0000000xxxx)

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 HW Test, SW 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

11.3.9 G2SB_CTX1_G2RAISE_0

Offset: 0x400d | Byte Offset: 0x10034 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE: Push back to read fifo when all commands in the channel are done.

11.3.10 G2SB_CTX1_G2HOSTSET_0

Offset: 0x400f | Byte Offset: 0x1003c | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

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

Bit	Reset	Description
		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)

11.3.11 G2SB_CTX1_G2HOSTFIFO_0

Offset: 0x4010 | Byte Offset: 0x10040 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

11.3.12 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 DDAReset 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. 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.

11.3.13 G2SB_CTX1_G2VDDAINI_0

Offset: 0x4012 | Byte Offset: 0x10048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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.

11.3.14 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')

Bit	Reset	Description
		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.

11.3.15 G2SB_CTX1_G2HDDAINILS_0

Offset: 0x4014 │ Byte Offset: 0x10050 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxx)

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.

11.3.16 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 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: 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 0x0CC. If 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

11.3.17 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

11.3.18 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

11.3.19 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, else the Key is set to 0.

StretchBLT, Color/Chroma Key Lower Limit Register

Offset: 0x4018 │ Byte Offset: 0x10060 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx)

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])

11.3.20 G2SB_CTX1_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0x4019 │ Byte Offset: 0x10064 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.3.21 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 422 format.

11.3.22 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 422 format.

11.3.23 G2SB_CTX1_G2SBFORMAT_0

Offset: 0x401c │ Byte Offset: 0x10070 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxx)

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 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
18 = RESERVED18
19 = RESERVED19
20 = RESERVED20
</td></tr>
</table>

Bit	Reset	Description
		21 = RESERVED21 22 = RESERVED22 23 = RESERVED23 24 = RESERVED24 25 = RESERVED25 26 = RESERVED26 27 = RESERVED27 28 = RESERVED28 29 = RESERVED29 30 = RESERVED30 31 = RESERVED31

11.3.24 G2SB_CTX1_G2CONTROLSB_0

Offset: 0x401d │ Byte Offset: 0x10074 │ Read/Write: R/W | Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000)

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 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

Bit	Reset	Description
		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 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 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 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.

11.3.25 G2SB_CTX1_G2CONTROLSECOND_0

Offset: 0x401e │ Byte Offset: 0x10078 │ Read/Write: R/W │ Reset: 0x00000000 (0b000000x00xxxx000xxxx000000000)

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 clock wise by 90 degrees) 101 = ROT_180 110 = ROT_270 (clock wise 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, sw 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 - dest base address DSTS - dest 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 2nd 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

Bit	Reset	Description
9	0x0	ALPSRCORDST: 32bits 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 1bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero. PL2BPP: ALPHA blending: Alpha 2bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero. PL4BPP: ALPHA blending: Alpha 4bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero. PL8BPP: ALPHA blending: Alpha 8bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1bit from source B5G5R5A1, alpha(MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4bits 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 hw by vcaa engine; not really a vcaa resolve) PLS8BPP: ALPHA blending: Alpha 8bits 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 8bits 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 128bit boundary. 2. Destination width has to be multiple of 4 pixels. 3. Source/Destination strides have to be multiple of 128bits. 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 1=enable 0= disable 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 1=enable 0= disable 1= enable 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 1=enable 0= disable 1= enable 0 = DISABLE 1 = ENABLE

11.3.26 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 doesn't 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 (ex: BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
  - ALPTYPE_PLS4BPP: xxxA_4444 (ex: BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
  - (other) : xxx_565 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32:
  - (all) : xxxA_8888 (ex: 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 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (ex: 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

srcColor format	srcCoverage format	legal dest format(s)
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();
```

SW *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 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 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 (0b000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: pattern Data Select: SRCSEL and PATSEL can't 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, don't 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 from frame buffer. Note, PACK is not officially supported when the surface is tiled.

Bit	Reset	Description
		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 16bytes 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 tell you 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 128bit/clock Some limitations with this mode:srcsld==1 rop==0xcc, no clipping, no transparency xdir==0, ydir==0, flip==0, xytdw==0Results 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 in executing (there are no other outstanding commands with same channel being executed)

Bit	Reset	Description
		0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

11.3.27 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.

11.3.28 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

11.3.29 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

11.3.30 G2SB_CTX1_G2CLIPRIGHTBOT_

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

11.3.31 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 (0bxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

11.3.32 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

11.3.33 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 16bytes aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

11.3.34 G2SB_CTX1_G2PATOS_0

Offset: 0x4027 | Byte Offset: 0x1009c | 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

11.3.35 G2SB_CTX1_G2PATBGC_0

Offset: 0x4028 | Byte Offset: 0x100a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

11.3.36 G2SB_CTX1_G2PATFGC_0

Offset: 0x4029 | Byte Offset: 0x100a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

11.3.37 G2SB_CTX1_G2PATKEY_0

Offset: 0x402a | Byte Offset: 0x100a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

11.3.38 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)

11.3.39 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

11.3.40 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)

11.3.41 G2SB_CTX1_G2DSTST_0

Offset: 0x402e | Byte Offset: 0x100b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

11.3.42 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 (0bxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

11.3.43 G2SB_CTX1_G2SRCPACK_SIZE_0

source data packed mode In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, it is required width/stride be 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

11.3.44 G2SB_CTX1_G2SRCBA_0

src 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)

11.3.45 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.)

11.3.46 G2SB_CTX1_G2SRCST_0

Offset: 0x4033 │ Byte Offset: 0x100cc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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.

11.3.47 G2SB_CTX1_G2SRCBGC_0

srcColors

Offset: 0x4034 | Byte Offset: 0x100d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

11.3.48 G2SB_CTX1_G2SRCFGC_0

Offset: 0x4035 | Byte Offset: 0x100d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

11.3.49 G2SB_CTX1_G2SRCKEY_0

Offset: 0x4036 | Byte Offset: 0x100d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

11.3.50 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

11.3.51 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

11.3.52 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

11.3.53 G2SB_CTX1_G2DSTPS_0

Offset: 0x403a | Byte Offset: 0x100e8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	DSTY
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

11.3.54 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 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

11.3.55 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 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 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)

11.3.56 G2SB_CTX1_G2LINESETTING_0

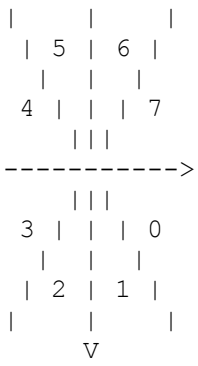
Line Methods

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

Bit	Reset	Description
		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: y major 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

11.3.57 G2SB_CTX1_G2LINEDELTAN_0



Offset: 0x403e | Byte Offset: 0x100f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

11.3.58 G2SB_CTX1_G2LINEDELTAM_0

Offset: 0x403f | Byte Offset: 0x100fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

11.3.59 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

11.3.60 G2SB_CTX1_G2LINELEN_0

Offset: 0x4041 | Byte Offset: 0x10104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

11.3.61 G2SB_CTX1_G2CSCFOURTH_0

Offset: 0x4042 | Byte Offset: 0x10108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.3.62 G2SB_CTX1_G2SRCST_B_0

Offset: 0x4043 | Byte Offset: 0x1010c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

11.3.63 G2SB_CTX1_G2UVSTRIDE_0

Offset: 0x4044 | Byte Offset: 0x10110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

11.3.64 G2SB_CTX1_G2CBDES2_0

Circular buffer controller 2

Offset: 0x4045 | Byte Offset: 0x10114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

11.3.65 G2SB_CTX1_G2TILEMODE_0

Offset: 0x4046 | Byte Offset: 0x10118 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxx0xxx0xxx0xxx0xxx0)

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

Bit	Reset	Description
		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

11.3.66 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

11.3.67 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

11.3.68 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

11.3.69 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

11.3.70 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

11.3.71 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

11.4 G2SB CTX 2 Registers

11.4.1 G2SB_CTX2_INCR_SYNCPT_0

Memory Controller Tiling definitions

Offset: 0x8000 | Byte Offset: 0x20000 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

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

11.4.2 G2SB_CTX2_INCR_SYNCPT_CNTRL_0

Offset: 0x8001 | Byte Offset: 0x20004 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx0)

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.

11.4.3 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.

11.4.4 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 (0bxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

11.4.5 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 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

11.4.6 G2SB_CTX2_G2TRIGGER1_0

Offset: 0x800a | Byte Offset: 0x20028 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

11.4.7 G2SB_CTX2_G2TRIGGER2_0

Offset: 0x800b | Byte Offset: 0x2002c | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

11.4.8 G2SB_CTX2_G2CMDSEL_0

Offset: 0x800c | Byte Offset: 0x20030 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxxxxxxx00x0x0000000xxxx)

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 HW Test, SW 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

11.4.9 G2SB_CTX2_G2RAISE_0

Offset: 0x800d | Byte Offset: 0x20034 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE: Push back to read fifo when all commands in the channel are done.

11.4.10 G2SB_CTX2_G2HOSTSET_0

Offset: 0x800f | Byte Offset: 0x2003c | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

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)

11.4.11 G2SB_CTX2_G2HOSTFIFO_0

Offset: 0x8010 | Byte Offset: 0x20040 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

11.4.12 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 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.

11.4.13 G2SB_CTX2_G2VDDAINI_0

Offset: 0x8012 | Byte Offset: 0x20048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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

Bit	Reset	Description
		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.

11.4.14 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.

11.4.15 G2SB_CTX2_G2HDDAINILS_0

Offset: 0x8014 | Byte Offset: 0x20050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx)

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.

11.4.16 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 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

11.4.17 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 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

11.4.18 G2SB_CTX2_G2CSCTHIRD_0

Offset: 0x8017 | Byte Offset: 0x2005c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

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

11.4.19 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 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

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])

11.4.20 G2SB_CTX2_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0x8019 | Byte Offset: 0x20064 | Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.4.21 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 422 format.

11.4.22 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 memory client will assemble YUV into one 422 format.

11.4.23 G2SB_CTX2_G2SBFORMAT_0

Offset: 0x801c | Byte Offset: 0x20070 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx)

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 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

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>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_*, 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</td></tr>
</table>

Bit	Reset	Description
		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

11.4.24 G2SB_CTX2_G2CONTROLSB_0

Offset: 0x801d │ Byte Offset: 0x20074 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000)

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

Bit	Reset	Description
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
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

Bit	Reset	Description
		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 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 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 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

Bit	Reset	Description
3	0x0	YUV422ROTATION: YUV422 planar rotated.

11.4.25 G2SB_CTX2_G2CONTROLSECOND_0

Offset: 0x801e │ Byte Offset: 0x20078 │ Read/Write: R/W │ Reset: 0x00000000 (0b000000x00xxxx000xxxx000000000)

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 clock wise 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) Max 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, sw 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 - dest base address DSTS - dest 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 2nd 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

Bit	Reset	Description
		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: 32bits 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 1bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero. PL2BPP: ALPHA blending: Alpha 2bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero. PL4BPP: ALPHA blending: Alpha 4bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero. PL8BPP: ALPHA blending: Alpha 8bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1bit from source B5G5R5A1, alpha(MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4bits 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 hw by vcaa engine; not really a vcaa resolve) PLS8BPP: ALPHA blending: Alpha 8bits 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 8bits 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 128bit boundary. 2. Destination width has to be multiple of 4 pixels. 3. Source/Destination strides have to be multiple of 128bits. 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 1=enable 0= disable 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 1=enable 0= disable 1= enable 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 1=enable 0= disable 1= enable 0 = DISABLE 1 = ENABLE

11.4.26 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 doesn't 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 (ex: BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
  - ALPTYPE_PLS4BPP: xxxA_4444 (ex: BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
  - (other) : xxx_565 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32:
  - (all) : xxxA_8888 (ex: 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 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (ex: 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();
```

SW *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 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 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 (0b000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: pattern Data Select: SRCSEL and PATSEL can't 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, don't 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.

Bit	Reset	Description
		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 16bytes 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 tell you 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 128bit/clock Some limitations with this mode:srcsld==1

Bit	Reset	Description
		rop==0xcc, no clipping, no transparency xdir==0, ydir==0, flip==0, xytdw==0Results 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 in executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

11.4.27 G2SB_CTX2_G2ROPFADE_0

Offset: 0x8020 | Byte Offset: 0x20080 | 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.

11.4.28 G2SB_CTX2_G2ALPHABLEND_0

Offset: 0x8021 | Byte Offset: 0x20084 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
31:24	X	ALPHA1V
23:16	X	ALPHA0V
8	0x0	ALPHAINV
7:0	X	ALPHA

11.4.29 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

11.4.30 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

11.4.31 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 (0bxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

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

11.4.32 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 16bytes aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

11.4.33 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,

Bit	Reset	Description
		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

11.4.34 G2SB_CTX2_G2PATBGC_0

Offset: 0x8028 | Byte Offset: 0x200a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

11.4.35 G2SB_CTX2_G2PATFGC_0

Offset: 0x8029 | Byte Offset: 0x200a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

11.4.36 G2SB_CTX2_G2PATKEY_0

Offset: 0x802a | Byte Offset: 0x200a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

11.4.37 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)

11.4.38 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

11.4.39 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)

11.4.40 G2SB_CTX2_G2DSTST_0

Offset: 0x802e | Byte Offset: 0x200b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

11.4.41 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 (0bxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

11.4.42 G2SB_CTX2_G2SRCPACK_SIZE_0

source data packed mode In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, it is required width/stride be 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

11.4.43 G2SB_CTX2_G2SRCBA_0

src 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)

11.4.44 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-bit, all 8 byte-positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there are no restrictions

Bit	Reset	Description
		on this value.)

11.4.45 G2SB_CTX2_G2SRCST_0

Offset: 0x8033 | Byte Offset: 0x200cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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.

11.4.46 G2SB_CTX2_G2SRCBGC_0

srcColors

Offset: 0x8034 | Byte Offset: 0x200d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

11.4.47 G2SB_CTX2_G2SRCFGC_0

Offset: 0x8035 | Byte Offset: 0x200d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

11.4.48 G2SB_CTX2_G2SRCKEY_0

Offset: 0x8036 | Byte Offset: 0x200d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

11.4.49 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

11.4.50 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

11.4.51 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

11.4.52 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

11.4.53 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 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

11.4.54 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 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 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)

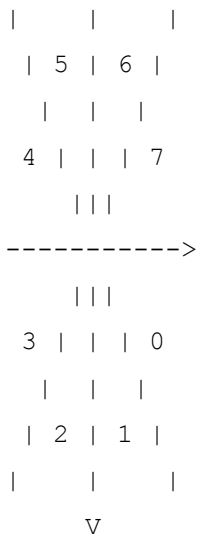
11.4.55 G2SB_CTX2_G2LINESETTING_0

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: y major 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

11.4.56 G2SB_CTX2_G2LINEDELTAN_0



Offset: 0x803e | Byte Offset: 0x200f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

11.4.57 G2SB_CTX2_G2LINEDELTAM_0

Offset: 0x803f | Byte Offset: 0x200fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

11.4.58 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

11.4.59 G2SB_CTX2_G2LINELEN_0

Offset: 0x8041 | Byte Offset: 0x20104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

11.4.60 G2SB_CTX2_G2CSCFOURTH_0

Offset: 0x8042 | Byte Offset: 0x20108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.4.61 G2SB_CTX2_G2SRCST_B_0

Offset: 0x8043 | Byte Offset: 0x2010c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

11.4.62 G2SB_CTX2_G2UVSTRIDE_0

Offset: 0x8044 | Byte Offset: 0x20110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

11.4.63 G2SB_CTX2_G2CBDES2_0

Circular buffer controller 2

Offset: 0x8045 | Byte Offset: 0x20114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

11.4.64 G2SB_CTX2_G2TILEMODE_0

Offset: 0x8046 | Byte Offset: 0x20118 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxx0xxx0xxx0xxx0xxx0)

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

11.4.65 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

11.4.66 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

11.4.67 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

11.4.68 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

11.4.69 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

11.4.70 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

11.5 G2SB CTX 3 Registers

11.5.1 G2SB_CTX3_INCR_SYNCPT_0

Memory Controller Tiling definitions

Offset: 0x1000 | Byte Offset: 0x4000 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:8	0x0	COND: Condition mapped from raise/wait 0 = IMMEDIATE

Bit	Reset	Description
		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

11.5.2 G2SB_CTX3_INCR_SYNCPT_CNTRL_0

Offset: 0x1001 │ Byte Offset: 0x4004 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxx0)

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.

11.5.3 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.

11.5.4 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 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

11.5.5 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 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

11.5.6 G2SB_CTX3_G2TRIGGER1_0

Offset: 0x100a | Byte Offset: 0x4028 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

11.5.7 G2SB_CTX3_G2TRIGGER2_0

Offset: 0x100b | Byte Offset: 0x402c | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

11.5.8 G2SB_CTX3_G2CMDSEL_0

Offset: 0x100c | Byte Offset: 0x4030 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxxxxxxx00x0x0000000xxxx)

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 HW Test, SW should not use this bit. 0 = DISABLE 1 = ENABLE
9:8	RW	0x0	G2OUTPUT: 00: to memory 01: to EPP 0 = MEMORY

Bit	R/W	Reset	Description
			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

11.5.9 G2SB_CTX3_G2RAISE_0

Offset: 0x100d │ Byte Offset: 0x4034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE: Push back to read fifo when all commands in the channel are done.

11.5.10 G2SB_CTX3_G2HOSTSET_0

Offset: 0x100f │ Byte Offset: 0x403c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000)

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)

11.5.11 G2SB_CTX3_G2HOSTFIFO_0

Offset: 0x1010 │ Byte Offset: 0x4040 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

11.5.12 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

Bit	Reset	Description
		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.

11.5.13 G2SB_CTX3_G2VDDAINI_0

Offset: 0x1012 | Byte Offset: 0x4048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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.

11.5.14 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.

11.5.15 G2SB_CTX3_G2HDDAINILS_0

Offset: 0x1014 | Byte Offset: 0x4050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx)

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

Bit	Reset	Description
		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.

11.5.16 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
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 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

Bit	Reset	Description
		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

11.5.17 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

11.5.18 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

11.5.19 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 (0bxxxxxxxxxxxxxxxxxxxxxxxx)

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

Bit	Reset	Description
		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])

11.5.20 G2SB_CTX3_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0x1019 │ Byte Offset: 0x4064 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.5.21 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 422 format.

11.5.22 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 422 format.

11.5.23 G2SB_CTX3_G2SBFORMAT_0

Offset: 0x101c │ Byte Offset: 0x4070 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxx)

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

Bit	Reset	Description
		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 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

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>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>

11.5.24 G2SB_CTX3_G2CONTROLSB_0

Offset: 0x101d │ Byte Offset: 0x4074 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000)

Bit	Reset	Description
31	0x0	DISDW: Output destination writes go either to image memory or the EPP.

Bit	Reset	Description
		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 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’.

Bit	Reset	Description
		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
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 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 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 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.

11.5.25 G2SB_CTX3_G2CONTROLSECOND_0

Offset: 0x101e │ Byte Offset: 0x4078 │ Read/Write: R/W │ Reset: 0x00000000 (0b000000x00xxxx000xxxx000000000)

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 clock wise by 90 degrees) 101 = ROT_180 110 = ROT_270 (clock wise 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

Bit	Reset	Description
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) Max 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, sw 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 - dest base address DSTS - dest 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 2nd 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: 32bits 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 1bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero PL2BPP: ALPHA blending: Alpha 2bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero PL4BPP: ALPHA blending: Alpha 4bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero PL8BPP: ALPHA blending: Alpha 8bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1bit from source B5G5R5A1, alpha (MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4bits 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 hw by vcaa engine; not really a vcaa resolve) PLS8BPP: ALPHA blending: Alpha 8bits from source/destination (decided by ALPSRCORDST), R8G8B8A8, alpha(MSB). DST: R8G8B8A8 VCAA: R8G8B8A8-

Bit	Reset	Description
		>R8G8B8A8, alpha has same blending method as RGB PLS8BX: ALPHA blending: Alpha 8bits 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 128bit boundary. 2. Destination width has to be multiple of 4 pixels. 3. Source/Destination strides have to be multiple of 128bits. 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 1=enable 0= disable 1= enable 0 = DISABLE 1 = ENABLE

11.5.26 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 doesn't 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:

a. ALPTYPE_PLS1BPP: xxxA_5551 (ex: BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])

b. ALPTYPE_PLS4BPP: xxxA_4444 (ex: BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])

c. (other) : xxx_565 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])

- color depth BPP32:

a. (all) : xxxA_8888 (ex: 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 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (ex: 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
- 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();
```

SW *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 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 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 (0b000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: pattern Data Select: SRCSEL and PATSEL can't 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, don't 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

Bit	Reset	Description
		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 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 16bytes 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 tell you 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 128bit/clock Some limitations with this mode:srcsld==1 rop==0xcc, no clipping, no transparencyxdir==0, ydir==0, flip==0, xytdw==0Results 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 in executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

11.5.27 G2SB_CTX3_G2ROPFADE_0

Offset: 0x1020 │ Byte Offset: 0x4080 │ 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.

11.5.28 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

11.5.29 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

11.5.30 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

11.5.31 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 (0bxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

11.5.32 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

11.5.33 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 16bytes aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

11.5.34 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

11.5.35 G2SB_CTX3_G2PATBGC_0

Offset: 0x1028 | Byte Offset: 0x40a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

11.5.36 G2SB_CTX3_G2PATFGC_0

Offset: 0x1029 | Byte Offset: 0x40a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

11.5.37 G2SB_CTX3_G2PATKEY_0

Offset: 0x102a | Byte Offset: 0x40a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

11.5.38 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)

11.5.39 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

11.5.40 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)

11.5.41 G2SB_CTX3_G2DSTST_0

Offset: 0x102e │ Byte Offset: 0x40b8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

11.5.42 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 (0bxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

11.5.43 G2SB_CTX3_G2SRCPACK_SIZE_0

source data packed mode In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, it is required width/stride be 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

11.5.44 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)

11.5.45 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-bit, all 8 byte-positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there are no restrictions on this value.)

11.5.46 G2SB_CTX3_G2SRCST_0

Offset: 0x1033 | Byte Offset: 0x40cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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.

11.5.47 G2SB_CTX3_G2SRCBGC_0

srcColors

Offset: 0x1034 | Byte Offset: 0x40d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

11.5.48 G2SB_CTX3_G2SRCFGC_0

Offset: 0x1035 | Byte Offset: 0x40d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

11.5.49 G2SB_CTX3_G2SRCKEY_0

Offset: 0x1036 | Byte Offset: 0x40d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

11.5.50 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

11.5.51 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

11.5.52 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 (bit7 if HLMONO==0, or bit0 if HLMONO==1) always expand to DSTX,DSTY

11.5.53 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

11.5.54 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 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

11.5.55 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 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 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)

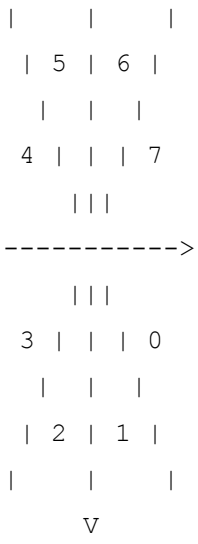
11.5.56 G2SB_CTX3_G2LINESETTING_0

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: y major 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

11.5.57 G2SB_CTX3_G2LINEDELTAN_0



Offset: 0x103e | Byte Offset: 0x40f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

11.5.58 G2SB_CTX3_G2LINEDELTAM_0

Offset: 0x103f | Byte Offset: 0x40fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

11.5.59 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

11.5.60 G2SB_CTX3_G2LINELEN_0

Offset: 0x1041 | Byte Offset: 0x4104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

11.5.61 G2SB_CTX3_G2CSCFOURTH_0

Offset: 0x1042 | Byte Offset: 0x4108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.5.62 G2SB_CTX3_G2SRCST_B_0

Offset: 0x1043 | Byte Offset: 0x410c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

11.5.63 G2SB_CTX3_G2UVSTRIDE_0

Offset: 0x1044 | Byte Offset: 0x4110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

11.5.64 G2SB_CTX3_G2CBDES2_0

Circular buffer controller 2

Offset: 0x1045 | Byte Offset: 0x4114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

11.5.65 G2SB_CTX3_G2TILEMODE_0

Offset: 0x1046 | Byte Offset: 0x4118 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxx0xxx0xxx0xxx0xxx0)

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

Bit	Reset	Description
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

11.5.66 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

11.5.67 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

11.5.68 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

11.5.69 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

11.5.70 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

11.5.71 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

11.6 G2SB CTX 4 Registers

11.6.1 G2SB_CTX4_INCR_SYNCPT_0

Memory Controller Tiling definitions

Offset: 0x5000 | Byte Offset: 0x14000 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

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

11.6.2 G2SB_CTX4_INCR_SYNCPT_CNTRL_0

Offset: 0x5001 | Byte Offset: 0x14004 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx0)

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.

11.6.3 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.

11.6.4 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 (0bxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

11.6.5 G2SB_CTX4_G2TRIGGER_0

Control methods, since they are not programmed often, put them is the beginning

Offset: 0x5009 | Byte Offset: 0x14024 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

11.6.6 G2SB_CTX4_G2TRIGGER1_0

Offset: 0x500a | Byte Offset: 0x14028 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

11.6.7 G2SB_CTX4_G2TRIGGER2_0

Offset: 0x500b │ Byte Offset: 0x1402c │ Read/Write: R/W │ Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

11.6.8 G2SB_CTX4_G2CMDSEL_0

Offset: 0x500c │ Byte Offset: 0x14030 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxx0xxxxxxxx00x0x0000000xxxx)

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 HW Test, SW 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

11.6.9 G2SB_CTX4_G2RAISE_0

Offset: 0x500d │ Byte Offset: 0x14034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE: Push back to read fifo when all commands in the channel are done.

11.6.10 G2SB_CTX4_G2HOSTSET_0

Offset: 0x500f | Byte Offset: 0x1403c | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

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)

11.6.11 G2SB_CTX4_G2HOSTFIFO_0

Offset: 0x5010 | Byte Offset: 0x14040 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

11.6.12 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 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.

11.6.13 G2SB_CTX4_G2VDDAINI_0

Offset: 0x5012 | Byte Offset: 0x14048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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.

11.6.14 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.

11.6.15 G2SB_CTX4_G2HDDAINILS_0

Offset: 0x5014 | Byte Offset: 0x14050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx)

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.

11.6.16 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 0x0CC. If 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

11.6.17 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

11.6.18 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

Bit	Reset	Description
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

11.6.19 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 (0bxxxxxxxxxxxxxxxxxxxxxxxx)

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])

11.6.20 G2SB_CTX4_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0x5019 │ Byte Offset: 0x14064 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.6.21 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 422 format.

11.6.22 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 422 format.

11.6.23 G2SB_CTX4_G2SBFORMAT_0

Offset: 0x501c │ Byte Offset: 0x14070 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxx)

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

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>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_*, 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</td></tr>
</table>

Bit	Reset	Description
		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

11.6.24 G2SB_CTX4_G2CONTROLSB_0

Offset: 0x501d │ Byte Offset: 0x14074 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000)

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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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.1 0 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 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 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 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.

Bit	Reset	Description
		0 = MULTIPLEX 1 = PLANAR
4	0x0	YUV422PLANAR: Source data is yuv422 planar valid when imode==PLANAR
3	0x0	YUV422ROTATION: YUV422 planar rotated.

11.6.25 G2SB_CTX4_G2CONTROLSECOND_0

Offset: 0x501e │ Byte Offset: 0x14078 │ Read/Write: R/W │ Reset: 0x00000000 (0b000000x00xxxx000xxxx000000000)

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 clock wise 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) Max 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, sw 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 - dest base address DSTS - dest 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 2nd 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 bits 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 1bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero. PL2BPP: ALPHA blending: Alpha 2bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero. PL4BPP: ALPHA blending: Alpha 4bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero. PL8BPP: ALPHA blending: Alpha 8bits/pixel from memory plane, B5G6R5 * B5G6R5 PL44BPP: ALPHA blending: Alpha 8bits/pixel from memory plane src*4bits+dst*4bits, B5G6R5 * B5G6R5 PLS1BPP: ALPHA blending: Alpha 1bit from source B5G5R5A1, alpha(MSB). Dest: B5G6R5. VCAA: reserved PLS4BPPAL: ALPHA blending: Alpha 4bits 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 hw by vcaa engine; not really a vcaa resolve) PLS8BPP: ALPHA blending: Alpha 8bits 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 8bits 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 128bit boundary. 2. Destination width has to be multiple of 4 pixels. 3. Source/Destination strides have to be multiple of 128bits. 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 1=enable 0= disable 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 1=enable 0= disable 1= enable 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 1=enable 0= disable 1= enable 0 = DISABLE 1 = ENABLE

11.6.26 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 doesn't 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 (ex: BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
  - ALPTYPE_PLS4BPP: xxxA_4444 (ex: BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
  - (other) : xxx_565 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32:
  - (all) : xxxA_8888 (ex: 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 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (ex: 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();
```

SW *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 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 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 (0b000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: pattern Data Select: SRCSEL and PATSEL can't 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, don't 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 S RCCD==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.

Bit	Reset	Description
		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 16bytes 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 tell you 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 128bit/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.

Bit	Reset	Description
1:0	0x0	CMDT: Command Type: 00=BitBlt 01=Line Draw 10=VCAA 11=reserved When the raise command is in executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

11.6.27 G2SB_CTX4_G2ROPFADE_0

Offset: 0x5020 | Byte Offset: 0x14080 | 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.

11.6.28 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

11.6.29 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

11.6.30 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

Bit	Reset	Description
14:0	X	CLIPR

11.6.31 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 (0bxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

11.6.32 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

11.6.33 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 16bytes aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

11.6.34 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.

Bit	Reset	Description
		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

11.6.35 G2SB_CTX4_G2PATBGC_0

Offset: 0x5028 | Byte Offset: 0x140a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

11.6.36 G2SB_CTX4_G2PATFGC_0

Offset: 0x5029 | Byte Offset: 0x140a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

11.6.37 G2SB_CTX4_G2PATKEY_0

Offset: 0x502a | Byte Offset: 0x140a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

11.6.38 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)

11.6.39 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

11.6.40 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)

11.6.41 G2SB_CTX4_G2DSTST_0

Offset: 0x502e │ Byte Offset: 0x140b8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

11.6.42 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 (0bxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

11.6.43 G2SB_CTX4_G2SRCPACK_SIZE_0

source data packed mode In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, it is required width/stride be 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

11.6.44 G2SB_CTX4_G2SRCBA_0

src 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)

11.6.45 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.)

11.6.46 G2SB_CTX4_G2SRCST_0

Offset: 0x5033 | Byte Offset: 0x140cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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.

11.6.47 G2SB_CTX4_G2SRCBGC_0

srcColors

Offset: 0x5034 | Byte Offset: 0x140d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

11.6.48 G2SB_CTX4_G2SRCFGC_0

Offset: 0x5035 | Byte Offset: 0x140d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

11.6.49 G2SB_CTX4_G2SRCKEY_0

Offset: 0x5036 | Byte Offset: 0x140d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

11.6.50 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

11.6.51 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

11.6.52 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

11.6.53 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

11.6.54 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 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

11.6.55 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 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 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)

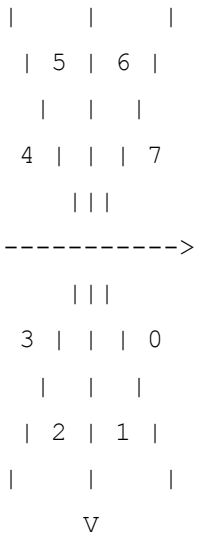
11.6.56 G2SB_CTX4_G2LINESETTING_0

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: y major 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

11.6.57 G2SB_CTX4_G2LINEDELTAN_0



Offset: 0x503e │ Byte Offset: 0x140f8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

11.6.58 G2SB_CTX4_G2LINEDELTAM_0

Offset: 0x503f | Byte Offset: 0x140fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

11.6.59 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

11.6.60 G2SB_CTX4_G2LINELEN_0

Offset: 0x5041 | Byte Offset: 0x14104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

11.6.61 G2SB_CTX4_G2CSCFOURTH_0

Offset: 0x5042 | Byte Offset: 0x14108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.6.62 G2SB_CTX4_G2SRCST_B_0

Offset: 0x5043 | Byte Offset: 0x1410c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

11.6.63 G2SB_CTX4_G2UVSTRIDE_0

Offset: 0x5044 | Byte Offset: 0x14110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

11.6.64 G2SB_CTX4_G2CBDES2_0

Circular buffer controller 2

Offset: 0x5045 | Byte Offset: 0x14114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

11.6.65 G2SB_CTX4_G2TILEMODE_0

Offset: 0x5046 | Byte Offset: 0x14118 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxx0xxx0xxx0xxx0xxx0)

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

11.6.66 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

11.6.67 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

11.6.68 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

11.6.69 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

11.6.70 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

11.6.71 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

11.7 G2SB CTX 5 Registers

11.7.1 G2SB_CTX5_INCR_SYNCPT_0

Memory Controller Tiling definitions

Offset: 0x2000 | Byte Offset: 0x8000 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

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

Bit	Reset	Description
		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

11.7.2 G2SB_CTX5_INCR_SYNCPT_CNTRL_0

Offset: 0x2001 │ Byte Offset: 0x8004 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxx0)

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.

11.7.3 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.

11.7.4 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 (0bxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

11.7.5 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 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

11.7.6 G2SB_CTX5_G2TRIGGER1_0

Offset: 0x200a | Byte Offset: 0x8028 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

11.7.7 G2SB_CTX5_G2TRIGGER2_0

Offset: 0x200b | Byte Offset: 0x802c | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

11.7.8 G2SB_CTX5_G2CMDSEL_0

Offset: 0x200c | Byte Offset: 0x8030 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxxxxxx00x0x0000000xxxx)

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 HW Test, SW 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

Bit	R/W	Reset	Description
4	RW	0x0	CBENABLE: circular buffer feature enable 0= disable 0 = DISABLE 1 = ENABLE
0	RO	X	SBOR2D: 0 = G2 1 = SB

11.7.9 G2SB_CTX5_G2RAISE_0

Offset: 0x200d │ Byte Offset: 0x8034 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE: Push back to read fifo when all commands in the channel are done.

11.7.10 G2SB_CTX5_G2HOSTSET_0

Offset: 0x200f │ Byte Offset: 0x803c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000)

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)

11.7.11 G2SB_CTX5_G2HOSTFIFO_0

Offset: 0x2010 │ Byte Offset: 0x8040 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

11.7.12 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 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.

11.7.13 G2SB_CTX5_G2VDDAINI_0

Offset: 0x2012 | Byte Offset: 0x8048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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.

11.7.14 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 DDAReset 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.

11.7.15 G2SB_CTX5_G2HDDAINILS_0

Offset: 0x2014 | Byte Offset: 0x8050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx)

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.

11.7.16 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 0x0CC. If 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

11.7.17 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

11.7.18 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

11.7.19 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), 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: 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])

11.7.20 G2SB_CTX5_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0x2019 | Byte Offset: 0x8064 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.7.21 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 422 format.

11.7.22 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 422 format.

11.7.23 G2SB_CTX5_G2SBFORMAT_0

Offset: 0x201c | Byte Offset: 0x8070 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx)

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

Bit	Reset	Description
		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]} 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}

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>
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>

11.7.24 G2SB_CTX5_G2CONTROLSB_0

Offset: 0x201d │ Byte Offset: 0x8074 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000)

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

Bit	Reset	Description
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 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].

Bit	Reset	Description
		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
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 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 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

Bit	Reset	Description
		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 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.

11.7.25 G2SB_CTX5_G2CONTROLSECOND_0

Offset: 0x201e │ Byte Offset: 0x8078 │ Read/Write: R/W │ Reset: 0x00000000 (0b000000x00xxxx000xxxx000000000)

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 clock wise 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) Max 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

Bit	Reset	Description
		surface dimension is not a multiple, sw 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 - dest base address DSTS - dest 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 2nd 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: 32bits 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 4bits 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 hw by 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 8bits 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 multiple of 4 pixels. 3. Source/Destination strides have to be multiple of 128bits. 0 = FIX 1 = PL1BPP 2 = PL2BPP 3 = PL4BPP

Bit	Reset	Description
		4 = PL8BPP 5 = PL44BPP 6 = PLS1BPP 7 = PLS4BPPAL 8 = PLS4BPP 9 = PLS8BPP 10 = PLS8BX 11 = PLS1BPPAL
3	0x0	BEWSWAP: Host port word swap 1=enable 0= disable 0 = DISABLE 1 = ENABLE
2	0x0	BEBSWAP: Host port byte swap 1=enable 0= disable 1= enable 0 = DISABLE 1 = ENABLE
1	0x0	BITSWAP: Host port bit swap 1=enable 0= disable 1= enable 0 = DISABLE 1 = ENABLE

11.7.26 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 doesn't 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 (ex: BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
  - ALPTYPE_PLS4BPP: xxxA_4444 (ex: BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])

- (other) : xxx_565 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32:
  - (all) : xxxA_8888 (ex: 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 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (ex: 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();
```

SW *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 bit 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 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 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 (0b000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: pattern Data Select: SRCSEL and PATSEL can't 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, don't 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

Bit	Reset	Description
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 16bytes 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 tell you 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

Bit	Reset	Description
		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 128bit/clockSome limitations with this mode:srcsld==1 rop==0xcc, no clipping, no transparencyxdir==0, ydir==0, flip==0, xytdw==0Results 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 in executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

11.7.27 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.

11.7.28 G2SB_CTX5_G2ALPHABLEND_0

Offset: 0x2021 | Byte Offset: 0x8084 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
31:24	X	ALPHA1V
23:16	X	ALPHA0V

Bit	Reset	Description
8	0x0	ALPHAINV
7:0	X	ALPHA

11.7.29 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

11.7.30 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

11.7.31 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 (0bxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

11.7.32 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

Bit	Reset	Description
15:0	X	PATMONOW: Mono data width in packed mode, byte

11.7.33 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 16bytes aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

11.7.34 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

11.7.35 G2SB_CTX5_G2PATBGC_0

Offset: 0x2028 | Byte Offset: 0x80a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

11.7.36 G2SB_CTX5_G2PATFGC_0

Offset: 0x2029 | Byte Offset: 0x80a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

11.7.37 G2SB_CTX5_G2PATKEY_0

Offset: 0x202a | Byte Offset: 0x80a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

11.7.38 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)

11.7.39 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

11.7.40 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)

11.7.41 G2SB_CTX5_G2DSTST_0

Offset: 0x202e | Byte Offset: 0x80b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

11.7.42 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 (0bxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

11.7.43 G2SB_CTX5_G2SRCPACK_SIZE_0

source data packed mode In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, it is required width/stride be 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
31:16	X	SRCMONOH: Packed mode - source mono data height in bytes
15:0	X	SRCMONOW: Packed mode - source mono data width in bytes

11.7.44 G2SB_CTX5_G2SRCBA_0

src 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)

11.7.45 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-bit, all 8 byte-positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there is no restrictions on this value.)

11.7.46 G2SB_CTX5_G2SRCST_0

Offset: 0x2033 | Byte Offset: 0x80cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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.

11.7.47 G2SB_CTX5_G2SRCBGC_0

srcColors

Offset: 0x2034 | Byte Offset: 0x80d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

11.7.48 G2SB_CTX5_G2SRCFGC_0

Offset: 0x2035 | Byte Offset: 0x80d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

11.7.49 G2SB_CTX5_G2SRCKEY_0

Offset: 0x2036 | Byte Offset: 0x80d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

11.7.50 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

11.7.51 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

11.7.52 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 (bit7 if HLMONO==0, or bit0 if HLMONO==1) alwaysexpand to DSTX,DSTY

11.7.53 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

11.7.54 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 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

11.7.55 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 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 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)

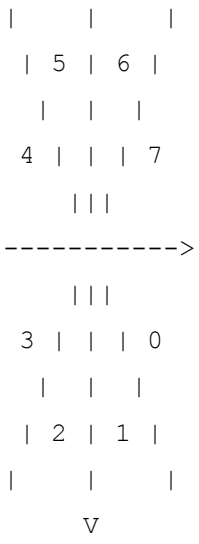
11.7.56 G2SB_CTX5_G2LINESETTING_0

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
24	X	MAJOR:0:xmajor 1: y major 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

11.7.57 G2SB_CTX5_G2LINEDELTAN_0



Offset: 0x203e | Byte Offset: 0x80f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

11.7.58 G2SB_CTX5_G2LINEDELTAM_0

Offset: 0x203f | Byte Offset: 0x80fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

11.7.59 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

11.7.60 G2SB_CTX5_G2LINELEN_0

Offset: 0x2041 | Byte Offset: 0x8104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

11.7.61 G2SB_CTX5_G2CSCFOURTH_0

Offset: 0x2042 | Byte Offset: 0x8108 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.7.62 G2SB_CTX5_G2SRCST_B_0

Offset: 0x2043 | Byte Offset: 0x810c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

11.7.63 G2SB_CTX5_G2UVSTRIDE_0

Offset: 0x2044 | Byte Offset: 0x8110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

11.7.64 G2SB_CTX5_G2CBDES2_0

Circular buffer controller 2

Offset: 0x2045 | Byte Offset: 0x8114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

11.7.65 G2SB_CTX5_G2TILEMODE_0

Offset: 0x2046 | Byte Offset: 0x8118 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxx0xxx0xxx0xxx0xxx0)

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

Bit	Reset	Description
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

11.7.66 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

11.7.67 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

11.7.68 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

11.7.69 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

11.7.70 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

11.7.71 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

11.8 G2SB CTX 6 Registers

11.8.1 G2SB_CTX6_INCR_SYNCPT_0

Memory Controller Tiling definitions

Offset: 0x6000 | Byte Offset: 0x18000 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

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

11.8.2 G2SB_CTX6_INCR_SYNCPT_CNTRL_0

Offset: 0x6001 | Byte Offset: 0x18004 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx0)

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.

11.8.3 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.

11.8.4 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 (0bxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

11.8.5 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 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

11.8.6 G2SB_CTX6_G2TRIGGER1_0

Offset: 0x600a | Byte Offset: 0x18028 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

11.8.7 G2SB_CTX6_G2TRIGGER2_0

Offset: 0x600b | Byte Offset: 0x1802c | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

11.8.8 G2SB_CTX6_G2CMDSEL_0

Offset: 0x600c | Byte Offset: 0x18030 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxxxxxxx00x0x0000000xxxx)

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 HW Test, SW 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

11.8.9 G2SB_CTX6_G2RAISE_0

Offset: 0x600d | Byte Offset: 0x18034 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE:Push back to read fifo when all commands in the channel are done.

11.8.10 G2SB_CTX6_G2HOSTSET_0

Offset: 0x600f | Byte Offset: 0x1803c | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

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

Bit	Reset	Description
		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)

11.8.11 G2SB_CTX6_G2HOSTFIFO_0

Offset: 0x6010 | Byte Offset: 0x18040 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

11.8.12 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 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.

11.8.13 G2SB_CTX6_G2VDDAINI_0

Offset: 0x6012 | Byte Offset: 0x18048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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 ValuesVertical 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.

11.8.14 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.

11.8.15 G2SB_CTX6_G2HDDAINILS_0

Offset: 0x6014 | Byte Offset: 0x18050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx)

Bit	Reset	Description
7:0	X	HDINI: Horizontal Scaling DDA Initial ValueHorizontal 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.

11.8.16 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 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 0x0CC. If 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

11.8.17 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
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

11.8.18 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 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

Bit	Reset	Description
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

11.8.19 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 (0bxxxxxxxxxxxxxxxxxxxxxxxx)

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])

11.8.20 G2SB_CTX6_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0x6019 │ Byte Offset: 0x18064 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.8.21 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 422 format.

11.8.22 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 422 format.

11.8.23 G2SB_CTX6_G2SBFORMAT_0

Offset: 0x601c │ Byte Offset: 0x18070 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxx)

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

<table>
<tr><th>Bit</th><th>Reset</th><th>Description</th></tr>
<tr><td></td><td></td><td>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_*, 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</td></tr>
</table>

Bit	Reset	Description
		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

11.8.24 G2SB_CTX6_G2CONTROLSB_0

Offset: 0x601d │ Byte Offset: 0x18074 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000)

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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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 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 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 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.

11.8.25 G2SB_CTX6_G2CONTROLSECOND_0

Offset: 0x601e │ Byte Offset: 0x18078 │ Read/Write: R/W │ Reset: 0x00000000 (0b000000x00xxxx000xxxx000000000)

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 clock wise 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) Max 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, sw 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 - dest base address DSTS - dest 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 2nd 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

Bit	Reset	Description
		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 hw by 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 8bits 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 multiple of 4 pixels. 3. Source/Destination strides have to be 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

11.8.26 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 doesn't 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 (ex: BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
  - ALPTYPE_PLS4BPP: xxxA_4444 (ex: BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
  - (other) : xxx_565 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32:
  - (all) : xxxA_8888 (ex: 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 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (ex: 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();
```

SW *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 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 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 c │ Byte Offset: 0x1807c │ Read/Write: R/W │ Reset: 0x00000000 (0b000000x00xx00xx0000000000000000)

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, don't 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 from frame buffer. Note, PACK is not officially supported when the surface is tiled. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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 16bytes 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 tell you 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 128bit/clockSome 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 in executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

11.8.27 G2SB_CTX6_G2ROPFADE_0

Offset: 0x6020 | Byte Offset: 0x18080 | 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.

11.8.28 G2SB_CTX6_G2ALPHABLEND_0

Offset: 0x6021 | Byte Offset: 0x18084 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx)

Bit	Reset	Description
31:24	X	ALPHA1V
23:16	X	ALPHA0V
8	0x0	ALPHAINV
7:0	X	ALPHA

11.8.29 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

11.8.30 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

11.8.31 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 (0bxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

11.8.32 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

11.8.33 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 16bytes aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

11.8.34 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

11.8.35 G2SB_CTX6_G2PATBGC_0

Offset: 0x6028 | Byte Offset: 0x180a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

11.8.36 G2SB_CTX6_G2PATFGC_0

Offset: 0x6029 | Byte Offset: 0x180a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

11.8.37 G2SB_CTX6_G2PATKEY_0

Offset: 0x602a | Byte Offset: 0x180a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

11.8.38 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)

11.8.39 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

11.8.40 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)

11.8.41 G2SB_CTX6_G2DSTST_0

Offset: 0x602e | Byte Offset: 0x180b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

11.8.42 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 (0bxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

11.8.43 G2SB_CTX6_G2SRCPACK_SIZE_0

Source data packed mode

In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, it is required width/stride be 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

11.8.44 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)

11.8.45 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-bit, all 8 byte-positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there are no restrictions on this value.)

11.8.46 G2SB_CTX6_G2SRCST_0

Offset: 0x6033 │ Byte Offset: 0x180cc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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

Bit	Reset	Description
		example, {0, 4} for any YcrCb formats, {0, 2, 4, 6} for RGB 16-bit format.

11.8.47 G2SB_CTX6_G2SRCBGC_0

srcColors

Offset: 0x6034 | Byte Offset: 0x180d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

11.8.48 G2SB_CTX6_G2SRCFGC_0

Offset: 0x6035 | Byte Offset: 0x180d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

11.8.49 G2SB_CTX6_G2SRCKEY_0

Offset: 0x6036 | Byte Offset: 0x180d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

11.8.50 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

11.8.51 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

11.8.52 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

Bit	Reset	Description
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) alwaysexpand to DSTX,DSTY

11.8.53 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

11.8.54 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 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

11.8.55 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 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 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)

11.8.56 G2SB_CTX6_G2LINESETTING_0

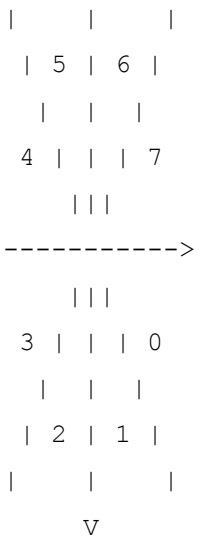
Line Methods

Offset: 0x603d | Byte Offset: 0x180f4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:29	X	OCTANTS:

Bit	Reset	Description
		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: y major 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

11.8.57 G2SB_CTX6_G2LINEDELTAN_0



Offset: 0x603e | Byte Offset: 0x180f8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

11.8.58 G2SB_CTX6_G2LINEDELTAM_0

Offset: 0x603f | Byte Offset: 0x180fc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

11.8.59 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

11.8.60 G2SB_CTX6_G2LINELEN_0

Offset: 0x6041 | Byte Offset: 0x18104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

11.8.61 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.

11.8.62 G2SB_CTX6_G2SRCST_B_0

Offset: 0x6043 | Byte Offset: 0x1810c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

11.8.63 G2SB_CTX6_G2UVSTRIDE_0

Offset: 0x6044 | Byte Offset: 0x18110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

11.8.64 G2SB_CTX6_G2CBDES2_0

Circular buffer controller 2

Offset: 0x6045 | Byte Offset: 0x18114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

11.8.65 G2SB_CTX6_G2TILEMODE_0

Offset: 0x6046 | Byte Offset: 0x18118 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxx0xxx0xxx0xxx0xxx0)

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

11.8.66 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

11.8.67 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

11.8.68 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

11.8.69 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

11.8.70 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

11.8.71 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

11.9 G2SB CTX 7 Registers

11.9.1 G2SB_CTX7_INCR_SYNCPT_0

Memory Controller Tiling definitions

Offset: 0xa000 | Byte Offset: 0x28000 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

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

Bit	Reset	Description
		15 = COND_15
7:0	0x0	INDX: syncpt index value

11.9.2 G2SB_CTX7_INCR_SYNCPT_CNTRL_0

Offset: 0xa001 │ Byte Offset: 0x28004 │ Read/Write: R/W │ Reset: 0x00000000 (0b0xxxxxx0)

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.

11.9.3 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.

11.9.4 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 (0bxxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
20	RW	X	CTX_VALID
19:16	RW	X	CURR_CHANNEL
9:0	RO	X	CURR_CLASS

11.9.5 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 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER

11.9.6 G2SB_CTX7_G2TRIGGER1_0

Offset: 0xa00a | Byte Offset: 0x28028 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER1

11.9.7 G2SB_CTX7_G2TRIGGER2_0

Offset: 0xa00b | Byte Offset: 0x2802c | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

Bit	Reset	Description
15:0	0x0	TRIGGER2

11.9.8 G2SB_CTX7_G2CMDSEL_0

Offset: 0xa00c | Byte Offset: 0x28030 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxxxxxxx00x0x0000000xxxx)

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 HW Test, SW 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

Bit	R/W	Reset	Description
			1 = SB

11.9.9 G2SB_CTX7_G2RAISE_0

Offset: 0xa00d | Byte Offset: 0x28034 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	RAISECHANNEL
4:0	X	RAISE:Push back to read fifo when all commands in the channel are done.

11.9.10 G2SB_CTX7_G2HOSTSET_0

Offset: 0xa00f | Byte Offset: 0x2803c | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

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)

11.9.11 G2SB_CTX7_G2HOSTFIFO_0

Offset: 0xa010 | Byte Offset: 0x28040 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	HOSTFIFODATA

11.9.12 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 DDAReset value: xxxx-xxxxhVertical 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. 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.

11.9.13 G2SB_CTX7_G2VDDAINI_0

Offset: 0xa012 | Byte Offset: 0x28048 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description

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.

11.9.14 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.

11.9.15 G2SB_CTX7_G2HDDAINILS_0

Offset: 0xa014 | Byte Offset: 0x28050 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx)

Bit	Reset	Description
7:0	X	HDINI: Horizontal Scaling DDA Initial ValueHorizontal 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.

11.9.16 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 0x0CC. If 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

11.9.17 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

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

11.9.18 G2SB_CTX7_G2CSCTHIRD_0

Offset: 0xa017 │ Byte Offset: 0x2805c │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

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

11.9.19 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 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

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])

11.9.20 G2SB_CTX7_G2CMKEYU_0

StretchBLT, Color/Chroma Key Upper Limit Register

Offset: 0xa019 │ Byte Offset: 0x28064 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

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.

11.9.21 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 422 format.

11.9.22 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 422 format.

11.9.23 G2SB_CTX7_G2SBFORMAT_0

Offset: 0xa01c │ Byte Offset: 0x28070 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxx)

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_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420</td><td>YUV 4:4:4</td><td>U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*</td></tr>
</table>
</td></tr>
</table>

Bit	Reset	Description
		U8Y8V8Y8_*, Y8U8Y8V8_*, V8Y8U8Y8_*, U8Y8V8Y8_*, YUV420 | YUV 4:4:4 | B5G6R5, B5G6R5BS, R8G8B8A8, B8G8R8A8 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

11.9.24 G2SB_CTX7_G2CONTROLSB_0

Offset: 0xa01d │ Byte Offset: 0x28074 │ Read/Write: R/W │ Reset: 0x00000000 (0b00x00000000000000x0xx0xx00000)

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)

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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 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 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

Bit	Reset	Description
		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 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.

11.9.25 G2SB_CTX7_G2CONTROLSECOND_0

Offset: 0xa01e │ Byte Offset: 0x28078 │ Read/Write: R/W │ Reset: 0x00000000 (0b000000x00xxxx000xxxx000000000)

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 clock wise by 90 degrees) 101 = ROT_180 110 = ROT_270 (clock wise 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, SW 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

Bit	Reset	Description
		DSTCD - bpp8, bpp16, bpp32 SRCBA - source base address SRCWIDTH - (width in pixels-1) SRCHEIGHT - (height in lines-1) SRCS - source stride DSTBA - dest base address DSTS - dest 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 2nd 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: 32bits 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 1bit/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL1BPP, DSTX[2:0] must be zero. PL2BPP: ALPHA blending: Alpha 2bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL2BPP, DSTX[1:0] must be zero. PL4BPP: ALPHA blending: Alpha 4bits/pixel from memory plane, B5G6R5 * B5G6R5 NOTE: when ALPTYPE is PL4BPP, DSTX[0:0] must be zero. PL8BPP: ALPHA blending: Alpha 8bits/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 hw by 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 8bits 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 128bit boundary. 2. Destination width has to be multiple of 4 pixels. 3. Source/Destination strides have to be multiple of 128bits. 0 = FIX 1 = PL1BPP 2 = PL2BPP 3 = PL4BPP 4 = PL8BPP 5 = PL44BPP 6 = PLS1BPP 7 = PLS4BPPAL 8 = PLS4BPP 9 = PLS8BPP 10 = PLS8BX

Bit	Reset	Description
		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

11.9.26 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 doesn't 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 (ex: BGRA_5551 = A[15] R[14:10] G[9:5] B[4:0])
  - ALPTYPE_PLS4BPP: xxxA_4444 (ex: BGRA_4444 = A[15:12] R[11:8] G[7:4] B[3:0])
  - (other) : xxx_565 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32:
  - (all) : xxxA_8888 (ex: 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 (ex: BGR_565 = R[15:11] G[10:5] B[4:0])
- color depth BPP32: xxxA_8888 (ex: 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();
```

SW *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 bit 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 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 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 (0b000000x00xx00xx0000000000000000)

Bit	Reset	Description
30	0x0	PATSEL: pattern Data Select: SRCSEL and PATSEL can't 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, don't 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.

Bit	Reset	Description
		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 16bytes 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 tell you 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 128bit/clockSome limitations with this mode:srcsld==1 rop==0xcc, no clipping, no transparencyxdir==0, ydir==0, flip==0, xytdw==0Results are undefined if the above limitations are not satisfied.
1:0	0x0	CMDT: Command Type: 00=BitBlt 01=Line Draw

Bit	Reset	Description
		10=VCAA 11=reserved When the raise command is in executing (there are no other outstanding commands with same channel being executed) 0 = BITBLT 1 = LINEDRAW 2 = VCAA 3 = RESERVED1

11.9.27 G2SB_CTX7_G2ROPFADE_0

Offset: 0xa020 | Byte Offset: 0x28080 | 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.

11.9.28 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

11.9.29 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

11.9.30 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

11.9.31 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 (0bxxxx)

Bit	Reset	Description
3:0	X	PATLNGAP: Packed mode, pattern data line gap. byte

11.9.32 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

11.9.33 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 16bytes aligned. }else{ Point to the first byte of the first pixel of pattern plane. }

11.9.34 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

11.9.35 G2SB_CTX7_G2PATBGC_0

Offset: 0xa028 | Byte Offset: 0x280a0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATBGC

11.9.36 G2SB_CTX7_G2PATFGC_0

Offset: 0xa029 | Byte Offset: 0x280a4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATFGC

11.9.37 G2SB_CTX7_G2PATKEY_0

Offset: 0xa02a | Byte Offset: 0x280a8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PATKEY

11.9.38 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)

11.9.39 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

11.9.40 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)

11.9.41 G2SB_CTX7_G2DSTST_0

Offset: 0xa02e | Byte Offset: 0x280b8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	DSTS: Destination Stride coordinate (bytes) with respect to DSTBA.

11.9.42 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 (0bxxxx)

Bit	Reset	Description
3:0	X	SRCLNGAP: Packed mode - source mono data line gap

11.9.43 G2SB_CTX7_G2SRCPACK_SIZE_0

source data packed mode In packed mode, SRCMONOW holds the horizontal size (bytes). If MONOH > 1, it is required width/stride be 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

11.9.44 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)

11.9.45 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-bit, all 8 byte-positions are valid in 4:2:0 mode. (Unlike multiplexed pixels format, there is no restrictions on this value.)

11.9.46 G2SB_CTX7_G2SRCST_0

Offset: 0xa033 | Byte Offset: 0x280cc | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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.

11.9.47 G2SB_CTX7_G2SRCBGC_0

srcColors

Offset: 0xa034 | Byte Offset: 0x280d0 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCBGC

11.9.48 G2SB_CTX7_G2SRCFGC_0

Offset: 0xa035 | Byte Offset: 0x280d4 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCFGC

11.9.49 G2SB_CTX7_G2SRCKEY_0

Offset: 0xa036 | Byte Offset: 0x280d8 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	SRCKEY

11.9.50 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

11.9.51 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

11.9.52 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 (bit7 if HLMONO==0, or bit0 if HLMONO==1) alwaysexpand to DSTX,DSTY

11.9.53 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

11.9.54 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 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

11.9.55 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 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 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)

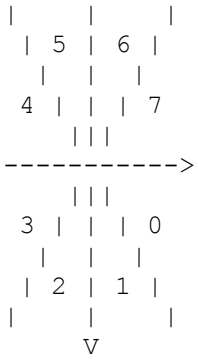
11.9.56 G2SB_CTX7_G2LINESETTING_0

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: y major 0 = XMAJOR 1 = YMAJOR
20:0	X	GAMMA

11.9.57 G2SB_CTX7_G2LINEDELTAN_0



Offset: 0xa03e │ Byte Offset: 0x280f8 │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAN

11.9.58 G2SB_CTX7_G2LINEDELTAM_0

Offset: 0xa03f │ Byte Offset: 0x280fc │ Read/Write: R/W │ Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
20:0	X	DELTAM

11.9.59 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

11.9.60 G2SB_CTX7_G2LINELEN_0

Offset: 0xa041 | Byte Offset: 0x28104 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	LINELEN

11.9.61 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.

11.9.62 G2SB_CTX7_G2SRCST_B_0

Offset: 0xa043 | Byte Offset: 0x2810c | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	SRCS_B: Source Stride B

11.9.63 G2SB_CTX7_G2UVSTRIDE_0

Offset: 0xa044 | Byte Offset: 0x28110 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:0	X	UVSTRIDE

11.9.64 G2SB_CTX7_G2CBDES2_0

Circular buffer controller 2

Offset: 0xa045 | Byte Offset: 0x28114 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxx)

Bit	Reset	Description
14:0	X	TOP_CBLINE: Circular buffer top clipping enabled, the first buffer line num

11.9.65 G2SB_CTX7_G2TILEMODE_0

Offset: 0xa046 | Byte Offset: 0x28118 | Read/Write: R/W | Reset: 0x00000000 (0b0xxx0xxx0xxx0xxx0xxx0xxx0)

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

11.9.66 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

11.9.67 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

11.9.68 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

11.9.69 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

11.9.70 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

11.9.71 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

11.10 G2SB Switch Registers

11.10.1 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, CPU received 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 (0b000)

Bit	Reset	Description
2	0x0	GR2D_IDLE
0	0x0	CTXSW_INT

11.10.2 G2SB_SWITCH_G2INTENABLE_0

Interrupt enable register

Offset: 0xf001 | Byte Offset: 0x3c004 | Read/Write: R/W | Reset: 0x00000000 (0b000)

Bit	Reset	Description
2	0x0	GR2D_IDLE_ENABLE
0	0x0	CTXSW_INT_ENABLE

11.10.3 G2SB_SWITCH_G2CURRENTCONTEXT_0

Offset: 0xf002 | Byte Offset: 0x3c008 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxx)

Bit	Reset	Description
7:0	X	CURR_CONTEXT

11.10.4 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 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
19:16	X	NEXT_CHANNEL
9:0	X	NEXT_CLASS

11.10.5 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 (0b00)

Bit	Reset	Description
1:0	0x0	DST_ADDR_IDX_INI:31:8 rw CLOCKFREEON init = 0

11.10.6 G2SB_SWITCH_G2GLOBALCONTROLB_0

Offset: 0xf005 | Byte Offset: 0x3c014 | Read/Write: R/W | Reset: 0x000000e1 (0b11100001)

Bit	Reset	Description
7:0	0xe1	OCTAN_BIAS

11.10.7 G2SB_SWITCH_G2WORKINGSTAT_0

Offset: 0xf006 | Byte Offset: 0x3c018 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxx)

Bit	Reset	Description
11:4	X	WORKING_CTX
3:0	X	WORKING_CHANNEL

11.10.8 G2SB_SWITCH_G2BUFTHRESHOLD_0

Offset: 0xf007 | Byte Offset: 0x3c01c | Read/Write: R/W | Reset: 0x00000000 (0b0000)

Bit	Reset	Description
3:0	0x0	BUFFER_COUNT_THRESHOLD

11.10.9 G2SB_SWITCH_CLKEN_OVERIDE_0

Offset: 0xf008 | Byte Offset: 0x3c020 | Read/Write: R/W | Reset: 0x00000000 (0b00000000000000000000000)

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
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

Bit	Reset	Description
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

11.10.10 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 clockenable fields of this register control the 2nd-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: 0xf009 │ Byte Offset: 0x3c024 │ Read/Write: R/W │ Reset: 0x00000000 (0b00xxxxxxxxxxxx0000)

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

11.10.11 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 (0b00110010)

Bit	Reset	Description
7:0	0x32	G2DW_WCOAL_TMVAL

11.10.12 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
15:8	0xff	CBR_G2PR2MC_DHYST_TM
7:0	0x6	CBR_G2PR2MC_HYST_REQ_TM

11.10.13 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

11.10.14 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

11.10.15 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

[THIS PAGE INTENTIONALLY LEFT BLANK]

12.0 EPP – ENCODER PRE-PROCESSOR

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.

12.1 Capabilities

The EPP receives input stream or surface data from VI or GR2D.

EPP receives input data on a 24-bit data bus with the format compatible to CSI output format. The input data formats are in six 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

```
                                    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 host and the same programmed value is used for all output pixels.

If EPP output processing is stalled due to stall from memory controllers or due to output trigger stall, EPP will propagate the stall condition to its input FIFO. If input FIFO becomes full, then EPP will not accept any more input data.

This stall may cause input data stream to be dropped (thrown away) if input stream cannot be stalled or if input stream comes from host via VI, then VI will stall host input stream processing.

12.2 Reset Sequence for EPP

1. Raise VI at end of EPP frame.

   If the first_output of VI is connected to 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 EPP, then RAISE_FRAME_2_VECTOR should be sent to register NV_VI_RAISE_VIP_FRAME_SECOND_OUTPUT_0 (offset 0x40).
2. Disable VI output to EPP by setting bits 4:2 to 0 of register NV_VI_VI_CORE_CONTROL_0 (offset 0x3)
3. Raise EPP at end of frame from host, NV_EPP_RAISE_FRAME_0 (offset 0x1a)
4. Read NV_EPP_CTXW_0 (0ffset 0) register, 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 MC.
6. Reset EPP by writing 0 to bit 6 of NV_HOST1X_ASYNC_RSTREG_0 (offset 0x14). Do read modify write.
7. Write 0 to NV_MC_CLIENT_HOTRESETN_0 (offset 0x19) bit 4 to clear EPP requests blocked inside 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 clear of EPP requests inside MC.
10. Write 1 to NV_MC_CLIENT_CTRL_0 bit 4 to unblock EPP request inside MC.
11. Bring EPP out of reset by writing 1 to bit 6 of NV_HOST1X_ASYNC_RSTREG_0 (offset 0x14). Do read modify write.
12. Program NV_EPP_CTXSW_0 (0x0) with previously saved EPP_CTXSW_DATA.
13. Program EPP registers
14. Enable encoder
15. Enable VI EPP output by programming bits 4:2 to the desired format of NV_VI_VI_CORE_CONTROL_0 (offset 0x3)

12.3 EPP Registers

12.3.1 EPP_INCR_SYNCPT_0

Offset: 0x0 | Byte Offset: 0x0 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000000)

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

12.3.2 EPP_INCR_SYNCPT_CNTRL_0

Offset: 0x1 | Byte Offset: 0x4 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx0)

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 soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

12.3.3 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.

12.3.4 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 (0b01)

Bit	Reset	Description
1	0x0	MPE
0	0x1	HOST

12.3.5 EPP_CTXSW_0

Context Switch Register. Should be common to all modules. Includes the current channel/class (which is writable by SW) 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). SW 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
9:0	RW	0x0	CURR_CLASS: Current working class

12.3.6 EPP_INTSTATUS_0

Interrupt Status. 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 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 EPP.
8	RW	X	SHORT_FRAME_INT: Short Frame Interrupt Status. If enabled, interrupt is generated when shorter than expected frame is detected. Input stream end-of-frame flag and OUTPUT_FRAME_SIZE is 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, interrupt is generated every time 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 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, interrupt is generated every time 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

12.3.7 EPP_EPP_CONTROL_0

EPP Control Register. This specifies processing control.

Offset: 0xb │ Byte Offset: 0x2c │ Read/Write: R/W │ Reset: 0x00000000 (0b0xx00000000000000000000)

Bit	Reset	Description
24	0x0	SW_FLOW_CONTROL: enable SW 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 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 output of EPP is used as input of JPEG encoder. JPEGE encoder can duplicate lines to ensure JPEGE has a full MCU to encode. If EPP Is used as input to MPEG encoder, output height of EPP has to be set to multiples of 16 for 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). Pixel duplication occurs prior to XY swap process therefore if XY_SWAP is set to ENABLE, the duplicated pixels will be placed on top and bottom of the image instead of left and right of the image and therefore may not help in JPEG encoding process. If image is to be rotated prior to JPEG encoding, it is preferred to do the rotation in the JPEG encoder rather than in EPP. 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. S/W 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 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 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

Bit	Reset	Description
		2 = YUYV : Y0UY1V, where Y0 is LSB and V is MSB 3 = YVYU : Y0VY1U, where Y0 is LSB and U is MSB
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 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

12.3.8 EPP_OUTPUT_FRAME_SIZE_0

Captured, input frame size

This specifies width and height of input frame with respect to size of active area to be processed by 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 (min 1 line).
15:0	0x0	FRAME_WIDTH: Frame width in pixels (min 16 pixels).

12.3.9 EPP_INPUT_FRAME_AOI_0

Input frame area of interest

This specifies position of the first pixel in the input frame to be processed by EPP. Together with 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 (min 0).
15:0	0x0	FRAME_HORI_OFFSET: Horizontal offset in term of pixel position (min 0).

12.3.10 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	XY swap (mirror on 315-degree diagonal)
1	0	1	270-degree rotation
1	1	0	90-degree rotation
1	1	1	XY swap and H,V flips (mirror on 45-degree diagonal)

Notes:

a. XY swap, if enabled, is performed first prior to H-flip and V-flip.

b. Output buffer start address should be programmed in consideration of the scan directions.

If xy-swap=0 (XY swap disabled):

OB0_Start_Address = OBSA + hsd * (width * bpp - 1) + vsd * (height - 1) * line_stride, where line_stride >= ((width * bpp) + 15) & 0xFFFFFFF0).

If xy-swap=1 (XY swap enabled):

OB0_Start_Address = OBSA + hsd * (height * bpp - 1) + vsd * (width - 1) * line_stride, where line_stride >= ((height * 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 byte 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 can NOT be performed.

Offset: 0xe │ Byte Offset: 0x38 │ Read/Write: R/W │ Reset: 0x00000000 (0b000)

Bit	Reset	Description
2	0x0	XY_SWAP: XY_SWAP IS NO LONGER SUPPORTED 0 = DISABLE : XY swap disabled. 1 = ENABLE : XY swap enabled.
1	0x0	VERT_DIR: Vertical scan direction. 0 = INCREASE : Increasing address. 1 = DECREASE : Decreasing address.

Bit	Reset	Description
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.

12.3.11 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.

12.3.12 EPP_OB0_BASE_ADDRESS_U_0

Output buffer 0 U base address. This is used to specify U buffer base address for planar output format. This is also used for semi-planar chroma buffer base address. This is not used for 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.

12.3.13 EPP_OB0_START_ADDRESS_V_0

Output buffer 0 V start address. This is used to specify V buffer start address for planar output format. This is not used for semi-planar and for 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.

12.3.14 EPP_OB0_BASE_ADDRESS_V_0

Output buffer 0 V base address. This is used to specify V buffer base address for planar output format. This is not used for semi-planar and for 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.

12.3.15 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

12.3.16 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.

12.3.17 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 (0bxxxxxxxxxxxxxxxxxxxxxxx00000000)

Bit	Reset	Description
28:16	X	OB0_V_SIZE: Output buffer vertical size. This specifies the number of lines. In the case of xyswap, vertical size should be programmed as vertical size before xyswap.
7:0	0x0	OB0_COUNT: Output buffer count. This specifies the number of buffers in output buffer set 0.

12.3.18 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 16-byte multiple so 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 16-byte multiple so 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.

12.3.19 EPP_CSC_RGB2Y_COEFF_0

RGB to Y coefficients

The coefficient for R component CSC_R2Y_COEFF is represented as Unsigned U0.8 (8-bit), the coefficient for G component CSC_G2Y_COEFF is represented as Unsigned U1.8 (9-bits), 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 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 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 (0b00000000000000000000000000)

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.

12.3.20 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 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 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 (0b00000000000000000000000000000000)

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.

12.3.21 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), 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 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 (0b00000000000000000000000000000000)

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.

12.3.22 EPP_CSC_YOFFSET_COEFF_0

Y Offset for the Color Space Conversion

Offset: 0x1e │ Byte Offset: 0x78 │ Read/Write: R/W │ Reset: 0x00000000 (0b00000000) │ 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.

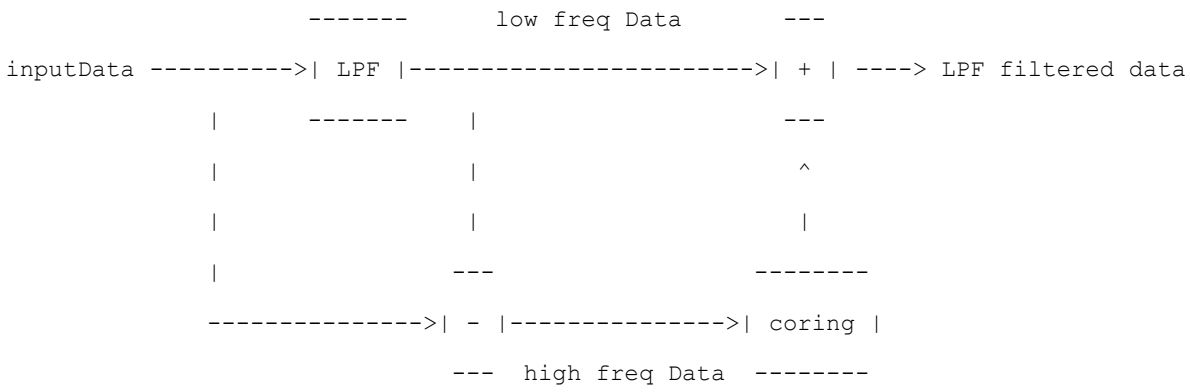
The recommended coefficients for luma pre-processing filter are: (0 1 4 6 4 1 0)/16. The filter coefficients are symmetric, therefore only four coefficients are programmed.

The recommended coefficient base is 16 and programmed to 4, in terms of power 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 power of 2. The sum of all coefficients should not exceed the coefficient base value.

The basic concept for this filter is:



12.3.23 EPP_FILTER_BOUND_0

Filter coring bound. This specifies min/max values for filter coring for both luma pre-processing filter and 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

12.3.24 EPP_FILTER_BASE_0

Filter coefficient base

Offset: 0x20 | Byte Offset: 0x80 | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxxxxxxx00000000) | 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.

12.3.25 EPP_PP_FILTER_COEF_0

Luma pre-processing filter coefficients. This 7-tap filter is symmetric therefore only four coefficients are programmed. 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 (0b00000000000000000000) | Default: 0x00031020

Bit	Reset	SW 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.

12.3.26 EPP_CHROMA_FILTER_COEF_0

Chroma 444-to-422 filter coefficients. This 7-tap filter is symmetric therefore only four coefficients are programmed. 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 (0b00000000000000000000) | 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.

12.3.27 EPP_ALFA_VALUE_0

Output alpha value. This is 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 (0b00000000)

Bit	Reset	Description
7:0	0x0	ALFA: Output alpha value.

12.3.28 EPP_U_LINE_BUFFER_ADDR_0

Chroma line buffer for U & 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 in 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.

12.3.29 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 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

12.3.30 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 maybe output trigger stall pending.

Offset: 0x26 | Byte Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0b00000xxxxxxxxxx00000)

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

12.3.31 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 maybe output trigger stall pending.

Offset: 0x27 │ Byte Offset: 0x9c │ Read/Write: R/W │ Reset: 0x00000000 (0b00000xxxxxxxxxx00000)

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

[THIS PAGE INTENTIONALLY LEFT BLANK]

13.0 KEYBOARD CONTROLLER

The Keyboard Controller (KBC) is used to interface a maximum of 24 pins with an external keypad. The number of pins used in the KBC depends on the external keypad row and column count, and is less in some Tegra® 3 package variations.

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 APB bus.

The typical software scanning flow is shown in Figure 18. Most of the tasks in the flowchart can be done in hardware, off-loading them from software. The three main operations done by KBC are:

- Keypad scan
- Debounce
- Control SM

In the presence of a KBC, the software flowchart for keyboard control is reduced to what is shown in Figure 19. 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 sizes of up to 16X8.
- Wake-up on interrupt mode
- Continuous polling mode
- Multi-key press support (dependency on keypad HW)
- 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. KBC is not aware of this difference and treats it as any other key; different modules in the Tegra 3 devices (e.g., Always ON) monitor this key when the device is powered off.

Row 0 should always be enabled. 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 18 Typical Software Scanning Flow

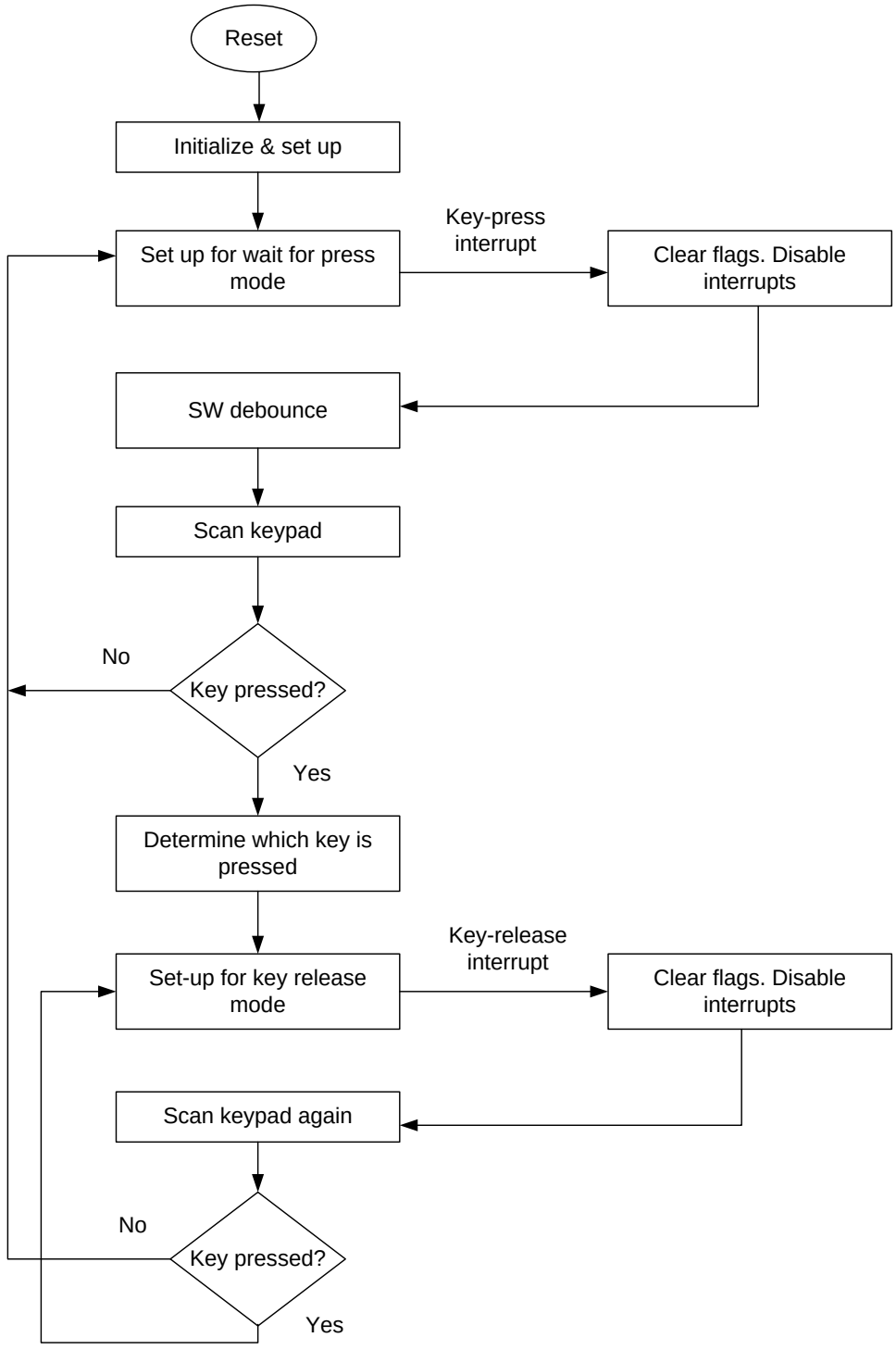


Figure 19 Software Flow with Hardware Keyboard Controller

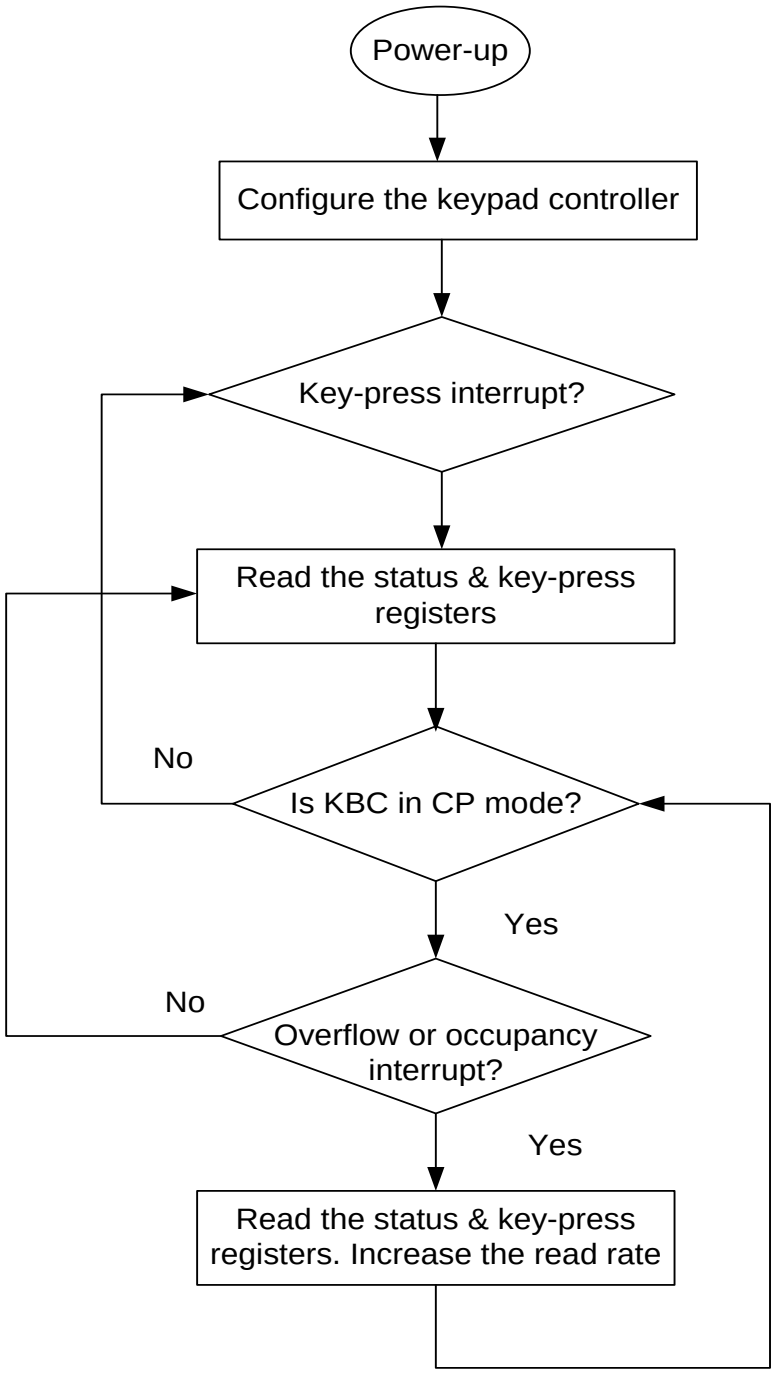
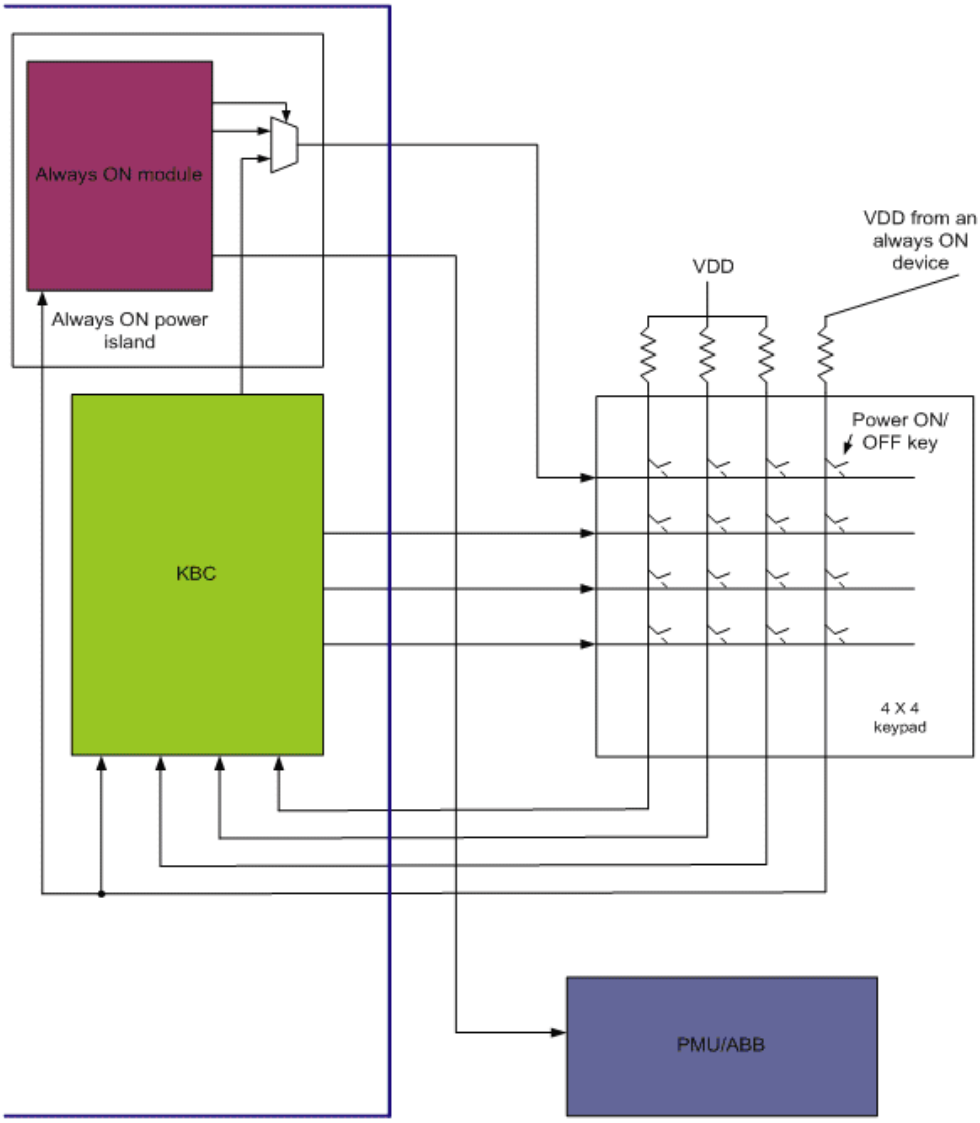


Figure 20 Power ON/OFF Key Example

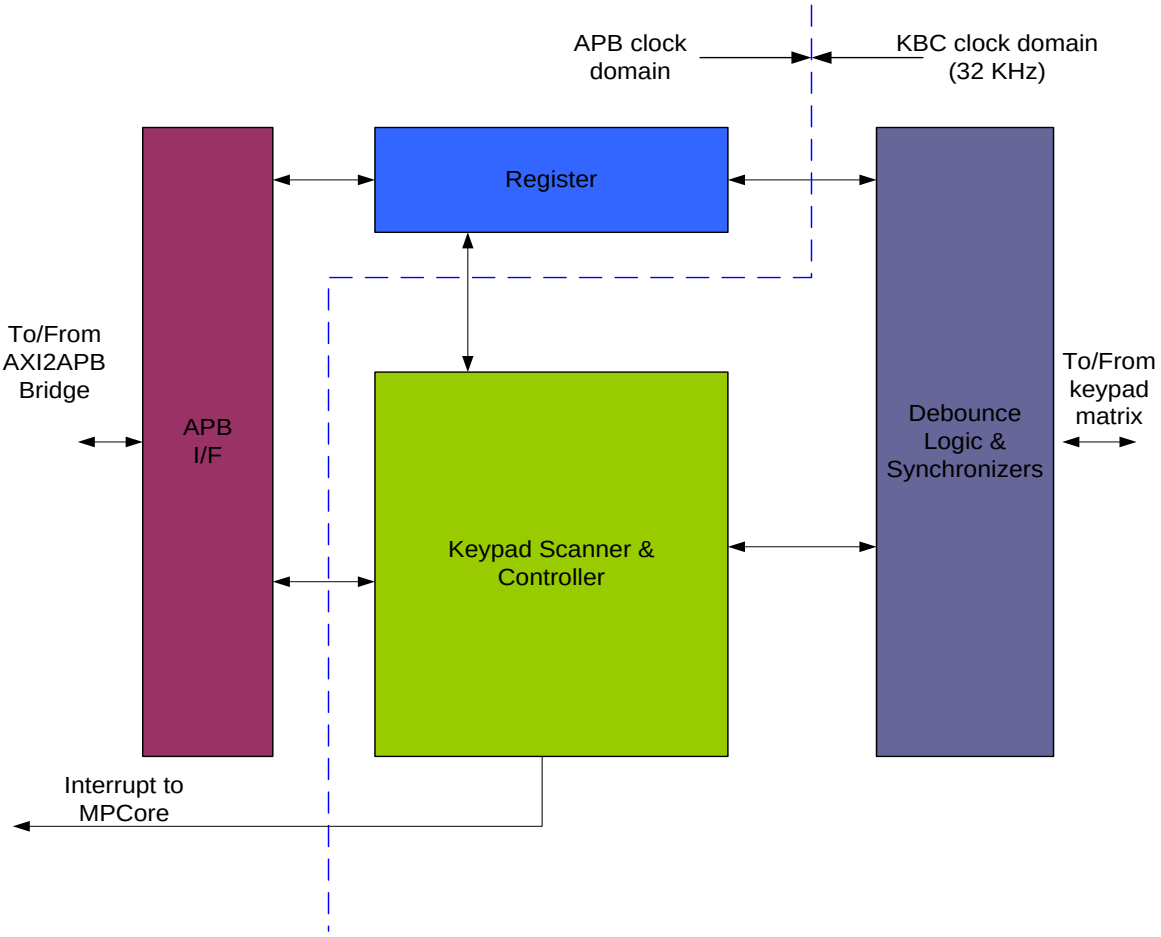


Clocking

- APB clock for APB interface and register sub-module
- KBC clock for keypad scanner and controller (32 KHz)

13.1 Functionality

Figure 21. 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 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

All the pins interfacing with keypad use de-bounce 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, 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 CP state, the delay between two consecutive full scans is RPT_DLY_VAL (in APBDEV_KBC_RPT_DLY register) cycles. In CP state, it scans all the rows one-by-one by driving one row low while other rows are configured as input so they are driven "Z". Then all the columns are sampled for each row scan.

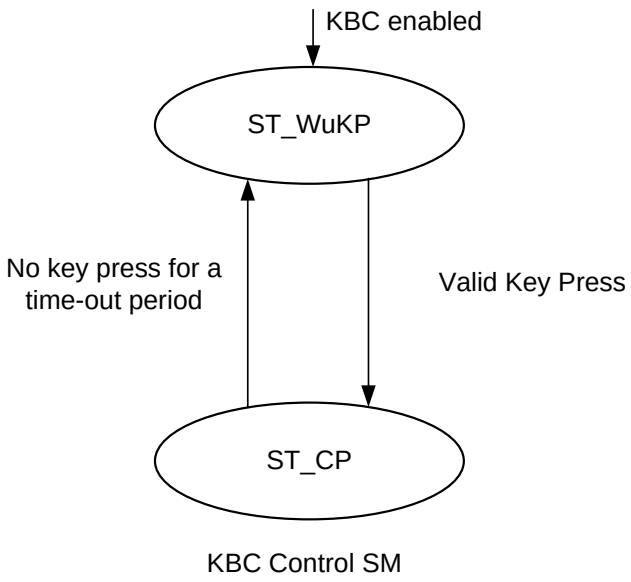
The driving of rows and sampling of columns is based on de-bounce cycle. The KBC cycle is 32 KHz (time period = 31.25 microseconds). So if de-bounce count is 4, one row scan takes 125 microseconds. One complete scan of all rows (16 max.) takes 2 milliseconds. In one second there can be 500 complete scans (if RPT_DLY_VAL = 0).

KBC supports up to eight simultaneous key-presses per complete scan. These eight key numbers are stored in the registers APBDEV_KBC_KP_ENT0 and APBDEV_KBC_KP_ENT1. 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 any wrong key-press more than once, software should see the AV_FIFO_CNT value in APBDEV_KBC_INT register. The entry registers give new key press information only when 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. Figure below shows KBC control the SM.

Figure 22 KBC Control Logic



13.2 Micro-Architecture

13.2.1 Keypad Matrix

The maximum supported keypad matrix size varies by the Tegra 3 package variation.

The KBC controller itself can support up to a 16x8 keypad matrix (16 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 eight events and a register stores 1 event.

A minimum of two rows must always be enabled.

13.2.2 Additional Interface Signals

kbc_interrrupt : An interrupt in the 32 KHz domain is sent to PMC and is used to wake up the system.

13.2.3 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 pullup/pulldown registers.

Rows are always output to the Tegra 3 and Columns are inputs. Row and Columns for the IO can be programmed by software as needed.

13.2.4 Ghost Key Detection

Ghost key detection is built-in to the way scanning interacts with keys that are simple switches. The solution is to have a diode in series with the key switch to avoid the ghost paths.

13.3 KBC Registers

13.3.1 APBDEV_KBC_CONTROL_0

This register is the main control register. It should be configured last after configuring all other settings.

KBC Control Register

Offset: 000h | Read/Write: R/W | Reset: 0b00100000000000000000

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 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: De-bounce count. This value sets the De-bounce FSM associated with each KBC input pin evaluate the input transitions 0 = No De-bounce N = N KBC clocks
3	0x0	FIFO_CNT_INT_EN: FIFO threshold count interrupt enable. Setting this bit will enable interrupt 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

13.3.2 APBDEV_KBC_INT_0

KBC Interrupt Status and Clear Register

Offset: 004h | Read/Write: R/W | Reset: 0bxxxxx000

Bit	R/W	Reset	Description
7:4	RO	X	AV_FIFO_CNT: FIFO occupancy count. Shows the number of unread registers. Read only.
3	RO	X	KBC_ST_STATUS: KBC status. Read only. 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 de-asserted 1 FIFO threshold count interrupt asserted (read) 1 Clear FIFO overflow interrupt (write)
1	RW	0x0	FIFO_OVF_INT_STATUS: FIFO overflow interrupt status. Writing '1' to this bit will clear the interrupt 0 FIFO overflow interrupt de-asserted 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 de-asserted 1 Key-press interrupt asserted (read) 1 Clear key-press interrupt (write)

13.3.3 APBDEV_KBC_ROW_CFG0_0

This register setting will take precedence in case same GPIO pin is configured in column register also. It does the mapping between GPIO pins and rows of keypad matrix. It configures for GPIO pin# 0 to 5.

First Row Configuration Register

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 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 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 keypad matrix. This bit overrides any setting done for pin# 3 in column configuration 0 = NOT_MAPPED 1 = MAPPED

Bit	Reset	Description
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 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 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
0	0x0	GPIO_0_ROW_EN: Indicates whether GPIO pin# 0 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 0 in column configuration 0 = NOT_MAPPED 1 = MAPPED

13.3.4 APBDEV_KBC_ROW_CFG1_0

This register setting takes precedence in case same GPIO pin is configured in column register also. It does the mapping between GPIO pins and rows of keypad matrix. It configures for GPIO pin# 6 to 11.

Second Row Configuration Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 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 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 keypad matrix. This bit overrides any setting done for pin# 9 in column configuration 0 = NOT_MAPPED

Bit	Reset	Description
		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 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 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 keypad matrix. This bit overrides any setting done for pin# 6 in column configuration 0 = NOT_MAPPED 1 = MAPPED

13.3.5 APBDEV_KBC_ROW_CFG2_0

This register setting takes precedence in case same GPIO pin is configured in column register also. It does the mapping between GPIO pins and rows of keypad matrix. It configures for GPIO pin# 12 to 17.

Third Row Configuration Register

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 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 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 keypad

Bit	Reset	Description
		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 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 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 keypad matrix. This bit overrides any setting done for pin# 12 in column configuration 0 = NOT_MAPPED 1 = MAPPED

13.3.6 APBDEV_KBC_ROW_CFG3_0

This register setting takes precedence in case same GPIO pin is configured in column register also. It does the mapping between GPIO pins & rows of keypad matrix. It configures for GPIO pin# 18 to 23.

Fourth Row Configuration Register

Offset: 014h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
29:26	0x0	GPIO_23_ROW_NUM: Mapping of GPIO pin# 23 to row number. Valid only if GPIO_21 is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
25	0x0	GPIO_23_ROW_EN: Indicates whether GPIO pin# 23 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 23 in column configuration 0 = NOT_MAPPED 1 = MAPPED
24:21	0x0	GPIO_22_ROW_NUM: Mapping of GPIO pin# 22 to row number. Valid only if GPIO_21 is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
20	0x0	GPIO_22_ROW_EN: Indicates whether GPIO pin# 22 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 22 in column configuration 0 = NOT_MAPPED 1 = MAPPED
19:16	0x0	GPIO_21_ROW_NUM: Mapping of GPIO pin# 21 to row number. Valid only if GPIO_21 is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
15	0x0	GPIO_21_ROW_EN: Indicates whether GPIO pin# 21 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 21 in column configuration

Bit	Reset	Description
		0 = NOT_MAPPED 1 = MAPPED
14:11	0x0	GPIO_20_ROW_NUM: Mapping of GPIO pin# 20 to row number. Valid only if GPIO_20 is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
10	0x0	GPIO_20_ROW_EN: Indicates whether GPIO pin# 20 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 20 in column configuration 0 = NOT_MAPPED 1 = MAPPED
9:6	0x0	GPIO_19_ROW_NUM: Mapping of GPIO pin# 19 to row number. Valid only if GPIO_19_ROW_EN is set. Indicates row number 0x0 = Row number 0 0xF = Row number 15
5	0x0	GPIO_19_ROW_EN: Indicates whether GPIO pin# 19 is mapped to any row of keypad matrix. This bit overrides any setting done for pin# 19 in column configuration 0 = NOT_MAPPED 1 = MAPPED
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 keypad matrix. This bit overrides any setting done for pin# 18 in column configuration 0 = NOT_MAPPED 1 = MAPPED

13.3.7 APBDEV_KBC_COL_CFG0_0

This register does the mapping between GPIO pins & columns of keypad matrix. It configures for GPIO pin# 0 to 7.

First Column Configuration Register

Offset: 018h | Read/Write: R/W | Reset: 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 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 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 keypad matrix. This bit should be set to '1' only when GPIO_5_ROW_EN in ROW_CFG0 is set to 0.

Bit	Reset	Description
		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 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 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
8	0x0	GPIO_2_COL_EN: Indicates whether GPIO pin# 2 is mapped to any column of 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 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 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

13.3.8 APBDEV_KBC_COL_CFG1_0

It does the mapping between GPIO pins & columns of keypad matrix. It configures for GPIO pin# 8 to 15.

Second Column Configuration Register

Offset: 01ch | Read/Write: R/W | Reset: 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 keypad

Bit	Reset	Description
		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 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 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 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
12	0x0	GPIO_11_COL_EN: Indicates whether GPIO pin# 11 is mapped to any column of 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 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 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

Bit	Reset	Description
		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 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

13.3.9 APBDEV_KBC_COL_CFG2_0

It does the mapping between GPIO pins & columns of keypad matrix. It configures for GPIO pin# 16 to 23.

Third Column Configuration Register

Offset: 020h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:29	0x0	GPIO_23_COL_NUM: Mapping of GPIO pin# 23 to column number. Valid only if GPIO_23 is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
28	0x0	GPIO_23_COL_EN: Indicates whether GPIO pin# 23 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_22 in ROW_CFG3 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
27:25	0x0	GPIO_22_COL_NUM: Mapping of GPIO pin# 22 to column number. Valid only if GPIO_22 is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
24	0x0	GPIO_22_COL_EN: Indicates whether GPIO pin# 22 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_22 in ROW_CFG3 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
23:21	0x0	GPIO_21_COL_NUM: Mapping of GPIO pin# 21 to column number. Valid only if GPIO_21 is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
20	0x0	GPIO_21_COL_EN: Indicates whether GPIO pin# 21 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_21 in ROW_CFG3 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
19:17	0x0	GPIO_20_COL_NUM: Mapping of GPIO pin# 20 to column number. Valid only if GPIO_20 is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
16	0x0	GPIO_20_COL_EN: Indicates whether GPIO pin# 20 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_20 in ROW_CFG3 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
15:13	0x0	GPIO_19_COL_NUM: Mapping of GPIO pin# 19 to column number. Valid only if GPIO_19_COL_EN is set. Indicates row number 0x0 = Column number 0 0x7 = Column number 7
12	0x0	GPIO_19_COL_EN: Indicates whether GPIO pin# 19 is mapped to any column of keypad matrix. This bit should be set to '1' only when GPIO_19_ROW_EN in ROW_CFG3 is set to 0. 0 = NOT_MAPPED 1 = MAPPED
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

Bit	Reset	Description
		number 7
8	0x0	GPIO_18_COL_EN: Indicates whether GPIO pin# 18 is mapped to any column of 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 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 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

13.3.10 APBDEV_KBC_TO_CNT_0

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).

Continuous Polling (CP) State Time-out Count Register

Offset: 024h | Read/Write: R/W | Reset: 0b00100111000100000000

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.

13.3.11 APBDEV_KBC_INIT_DLY_0

This register holds the initial delay for scan when KBC goes from WuKP mode to CP mode.

Offset: 028h | Read/Write: R/W | Reset: 0b00000000010000000000

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.

13.3.12 APBDEV_KBC_RPT_DLY_0

This register holds the repeat scan delay. This is the delay between two successive scans in CP mode.

Offset: 02ch | Read/Write: R/W | Reset: 0b00000000010000000000

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.

13.3.13 APBDEV_KBC_KP_ENT0_0

This register holds the key numbers of the keys which have been pressed.

It holds the first four entries

NOTE: Both the entry registers should be read consecutively. A read of first register changes the FIFO read pointer

First Key-Press Entries Register

Offset: 030h | Read/Write: RO | Reset: 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
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

13.3.14 APBDEV_KBC_KP_ENT1_0

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 the entry registers should be read consecutively. A read of first register changes the FIFO read pointer

Second Key-press Entries Register

Offset: 034h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	KP_NEW_ENT7: Indicates whether eight entry is valid or not 0x0 Entry not valid 0x1 Valid entry
30:27	X	KP_ROW_NUM_ENT7: Row number for eight key. 0x0 = Row number 0 0xF = Row number 15
26:24	X	KP_COL_NUM_ENT7: Column number for eight 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

13.3.15 APBDEV_KBC_ROW0_MASK_0

Offset: 038h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW0_COL7_MASK_ENABLE: Disable row0 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW0_COL6_MASK_ENABLE: Disable row0 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW0_COL5_MASK_ENABLE: Disable row0 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
4	0x0	KBC_ROW0_COL4_MASK_ENABLE: Disable row0 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW0_COL3_MASK_ENABLE: Disable row0 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW0_COL2_MASK_ENABLE: Disable row0 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW0_COL1_MASK_ENABLE: Disable row0 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW0_COL0_MASK_ENABLE: Disable row0 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.16 APBDEV_KBC_ROW1_MASK_0

Offset: 03ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW1_COL7_MASK_ENABLE: Disable row1 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW1_COL3_MASK_ENABLE: Disable row1 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE

Bit	Reset	Description
		1 = DISABLE
0	0x0	KBC_ROW1_COL0_MASK_ENABLE: Disable row1 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.17 APBDEV_KBC_ROW2_MASK_0

Offset: 040h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW2_COL7_MASK_ENABLE: Disable row2 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.18 APBDEV_KBC_ROW3_MASK_0

Offset: 044h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW3_COL7_MASK_ENABLE: Disable row3 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
6	0x0	KBC_ROW3_COL6_MASK_ENABLE: Disable row3 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.19 APBDEV_KBC_ROW4_MASK_0

Offset: 048h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW4_COL7_MASK_ENABLE: Disable row4 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW4_COL3_MASK_ENABLE: Disable row4 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE

Bit	Reset	Description
		1 = DISABLE
2	0x0	KBC_ROW4_COL2_MASK_ENABLE: Disable row4 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.20 APBDEV_KBC_ROW5_MASK_0

Offset: 04ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW5_COL7_MASK_ENABLE: Disable row5 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.21 APBDEV_KBC_ROW6_MASK_0

Offset: 050h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW6_COL7_MASK_ENABLE: Disable row6 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.22 APBDEV_KBC_ROW7_MASK_0

Offset: 054h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW7_COL7_MASK_ENABLE: Disable row7 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
4	0x0	KBC_ROW7_COL4_MASK_ENABLE: Disable row7 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW7_COL3_MASK_ENABLE: Disable row7 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.23 APBDEV_KBC_ROW8_MASK_0

Offset: 058h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW8_COL7_MASK_ENABLE: Disable row8 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE

Bit	Reset	Description
		1 = DISABLE
0	0x0	KBC_ROW8_COL0_MASK_ENABLE: Disable row8 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.24 APBDEV_KBC_ROW9_MASK_0

Offset: 05ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW9_COL7_MASK_ENABLE: Disable row9 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.25 APBDEV_KBC_ROW10_MASK_0

Offset: 060h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW10_COL7_MASK_ENABLE: Disable row10 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
6	0x0	KBC_ROW10_COL6_MASK_ENABLE: Disable row10 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW10_COL3_MASK_ENABLE: Disable row10 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col 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/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.26 APBDEV_KBC_ROW11_MASK_0

Offset: 064h | Read/Write: R/W | Secure: Unprotected | Reset: 0b00000000 | Default: 0000.0000

Bit	Reset	Description
7	0x0	KBC_ROW11_COL7_MASK_ENABLE: Disable row11 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW11_COL6_MASK_ENABLE: Disable row11 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW11_COL5_MASK_ENABLE: Disable row11 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW11_COL4_MASK_ENABLE: Disable row11 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW11_COL3_MASK_ENABLE: Disable row11 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE

Bit	Reset	Description
		1 = DISABLE
2	0x0	KBC_ROW11_COL2_MASK_ENABLE: Disable row11 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW11_COL1_MASK_ENABLE: Disable row11 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW11_COL0_MASK_ENABLE: Disable row11 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.27 APBDEV_KBC_ROW12_MASK_0

Offset: 068h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW12_COL7_MASK_ENABLE: Disable row12 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW12_COL6_MASK_ENABLE: Disable row12 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW12_COL5_MASK_ENABLE: Disable row12 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW12_COL4_MASK_ENABLE: Disable row12 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW12_COL3_MASK_ENABLE: Disable row12 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW12_COL2_MASK_ENABLE: Disable row12 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW12_COL1_MASK_ENABLE: Disable row12 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW12_COL0_MASK_ENABLE: Disable row12 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.28 APBDEV_KBC_ROW13_MASK_0

Offset: 06ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW13_COL7_MASK_ENABLE: Disable row13 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW13_COL6_MASK_ENABLE: Disable row13 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW13_COL5_MASK_ENABLE: Disable row13 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW13_COL4_MASK_ENABLE: Disable row13 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW13_COL3_MASK_ENABLE: Disable row13 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW13_COL2_MASK_ENABLE: Disable row13 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW13_COL1_MASK_ENABLE: Disable row13 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW13_COL0_MASK_ENABLE: Disable row13 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.29 APBDEV_KBC_ROW14_MASK_0

Offset: 070h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW14_COL7_MASK_ENABLE: Disable row14 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW14_COL6_MASK_ENABLE: Disable row14 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW14_COL5_MASK_ENABLE: Disable row14 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
4	0x0	KBC_ROW14_COL4_MASK_ENABLE: Disable row14 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW14_COL3_MASK_ENABLE: Disable row14 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW14_COL2_MASK_ENABLE: Disable row14 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
1	0x0	KBC_ROW14_COL1_MASK_ENABLE: Disable row14 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW14_COL0_MASK_ENABLE: Disable row14 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

13.3.30 APBDEV_KBC_ROW15_MASK_0

Offset: 074h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	KBC_ROW15_COL7_MASK_ENABLE: Disable row15 col7 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
6	0x0	KBC_ROW15_COL6_MASK_ENABLE: Disable row15 col6 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
5	0x0	KBC_ROW15_COL5_MASK_ENABLE: Disable row15 col5 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
4	0x0	KBC_ROW15_COL4_MASK_ENABLE: Disable row15 col4 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
3	0x0	KBC_ROW15_COL3_MASK_ENABLE: Disable row15 col3 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
2	0x0	KBC_ROW15_COL2_MASK_ENABLE: Disable row15 col2 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
1	0x0	KBC_ROW15_COL1_MASK_ENABLE: Disable row15 col1 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE
0	0x0	KBC_ROW15_COL0_MASK_ENABLE: Disable row15 col0 when system is in suspend/deep sleep mode 1 Disable row/col pair 0 Enable row/col pair 0 = ENABLE 1 = DISABLE

14.0 GPIO CONTROLLER

The Tegra® 3 devices provide a large number of potential general-purpose I/O (GPIO) pins. The actual number of available GPIO pins depends on the system pin-mux configuration. These pins are grouped as 31 8-bit ports. The ports are grouped into 8 GPIO controllers containing four 8-bit ports each. All these port pins are designed with multiplexing logic, so they can be used as either a dedicated Special Function I/O (SFIO) or a GPIO.

Features

- Multiplexed GPIO/SFIO functionality
- Protection against meta-stability for GPIO inputs (back-to-back D flip-flops)
- Interrupts are programmable on each individual GPIO pin
- Interrupt levels also programmable
- All pin configurations are independent of the other pins
- Masked-write registers

> **Note:** Some pin attributes that affect GPIO operation, including Tristate, Pullup/down, drive/schmidt control, are on Pin Group or Pin Config (group) granularity. These controls are outside the GPIO block.

Each port has a set of configuration registers, which are used for:

- GPIO/SFIO selection
- Enabling outputs
- Driving the output pin with a HIGH or LOW
- Receiving the input when in GPIO mode
- Enabling interrupts
- Checking status
- Clearing interrupts

14.1 Functionality

There are 244 pins that can function as either multiplexed Special Function I/O (SFIO) pins or General-Purpose I/O (GPIO) pins. In Special function mode, these pins are driven by the pinmux controller, where, each pin is driven by a special function controller selected based on the pinmux option (primary, alternate-1, alternate-2 and alternate-3).

This multiplexing architecture helps to support a much higher number of IOs on a minimal number of pads on the chip. There are additional pins too, that function as SFIOs only, without sharing the GPIO function. GPIO programmability gives more flexibility to the firmware. In addition to this, the GPIO architecture gives Firmware the flexibility of receiving any interrupt from an external requestor on any GPIO pin.

GPIO Pins

Each GPIO controller consists of 4 GPIO ports. Each port has a set of registers capable of controlling 8 GPIO pins. The ports for the first controller are named A, B, C, and D. The ports for the second controller are named E, F, G, and H. To find the offset of a particular register for a particular port, add the port offset from Table 29 to the register's offset. For example, the offset of GPIO Port R's GPIO_OUT register is 0x404 (Port Offset) + 0x20 (Register Offset) = 0x424.

Table 29: GPIO Controllers, Ports, and Pins

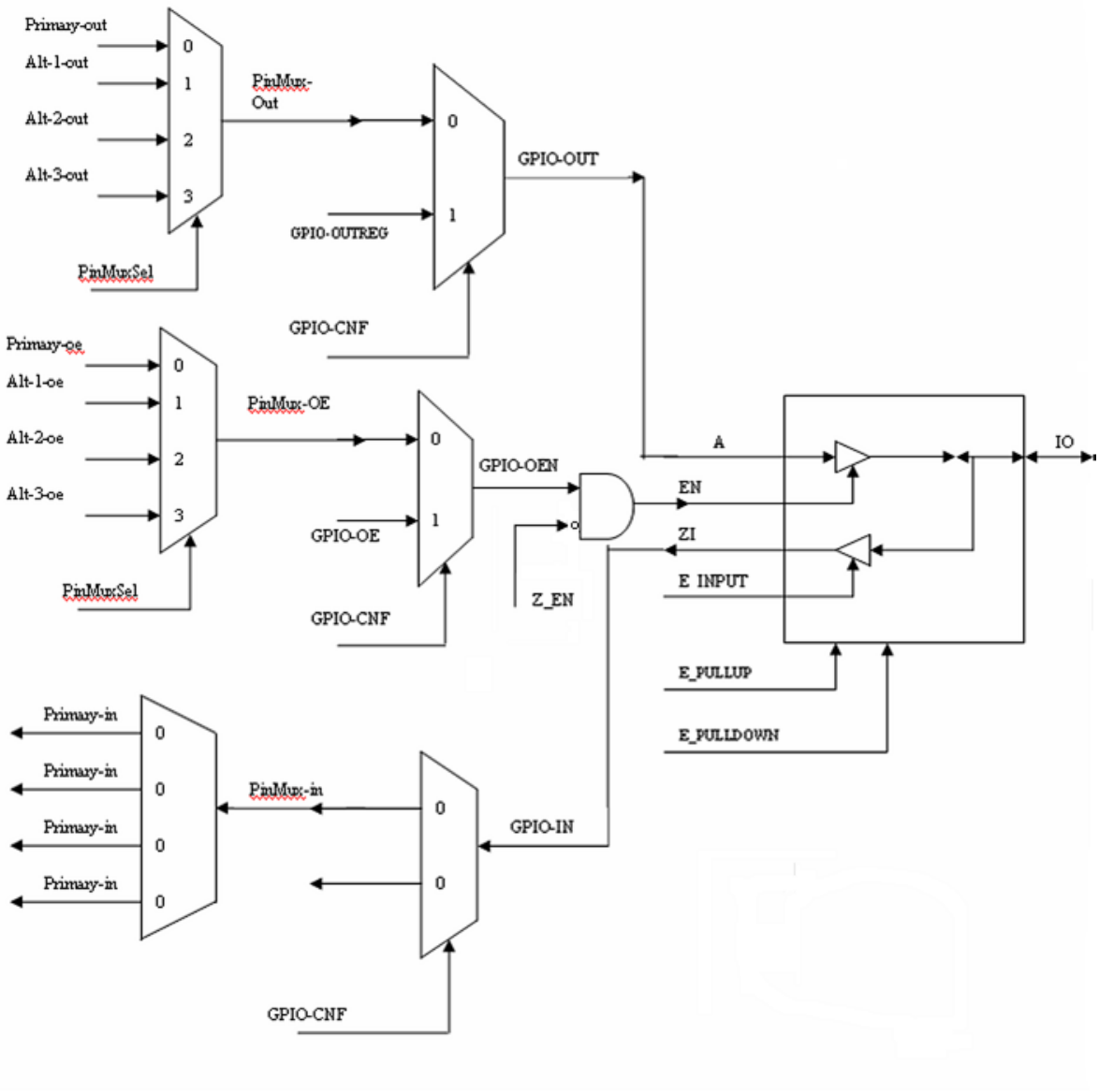
Controller	Port	Port Offset	Pins
GPIO1	Port A	0x000	GPIO_PA0 - GPIO_PA7
	Port B	0x004	GPIO_PB0 - GPIO_PB7
	Port C	0x008	GPIO_PC0 - GPIO_PC7
	Port D	0x00C	GPIO_PD0 - GPIO_PD7
GPIO2	Port E	0x100	GPIO_PE0 - GPIO_PE7
	Port F	0x104	GPIO_PF0 - GPIO_PF7
	Port G	0x108	GPIO_PG0 - GPIO_PG7
	Port H	0x10c	GPIO_PH0 - GPIO_PH7
GPIO3	Port I	0x200	GPIO_PI0 - GPIO_PI7
	Port J	0x204	GPIO_PJ0 - GPIO_PJ7
	Port K	0x208	GPIO_PK0 - GPIO_PK7
	Port L	0x20C	GPIO_PL0 - GPIO_PL7
GPIO4	Port M	0x300	GPIO_PM0 - GPIO_PM7
	Port N	0x304	GPIO_PN0 - GPIO_PN7
	Port O	0x308	GPIO_PO0 - GPIO_PO7
	Port P	0x30C	GPIO_PP0 - GPIO_PP7
GPIO5	Port Q	0x400	GPIO_PQ0 - GPIO_PQ7
	Port R	0x404	GPIO_PR0 - GPIO_PR7
	Port S	0x408	GPIO_PS0 - GPIO_PS7
	Port T	0x40c	GPIO_PT0 - GPIO_PT7
GPIO6	Port U	0x500	GPIO_PU0 - GPIO_PU7
	Port V	0x504	GPIO_PV0 - GPIO_PV7
	Port W	0x508	GPIO_PW0 - GPIO_PW7
	Port X	0x50C	GPIO_PX0 - GPIO_PX7
GPIO7	Port Y	0x600	GPIO_PY0 - GPIO_PY7
	Port Z	0x604	GPIO_PZ0 - GPIO_PZ7
	Port AA	0x608	GPIO_PAA0 - GPIO_PAA7
	Port BB	0x60C	GPIO_PBB0 - GPIO_PBB7
GPIO8	Port CC	0x700	GPIO_PCC0 - GPIO_PCC7
	Port DD	0x704	GPIO_PDD0 - GPIO_PDD7
	Port EE	0x708	GPIO_PEE0 - GPIO_PEE7

Use the appropriate GPIO_CNF register to designate a pin to act either as a GPIO as an SFIO. Setting a pin's CNF bit to 1 configures the pin to be used as a GPIO and prevents the pin from being used as for its Primary, Alternate 1, Alternate 2, or Alternate 3 SFIO function. Setting the CNF bit to 0 allows the port to be used as an SFIO.

For each pin designated to act as a GPIO, the corresponding bit in the appropriate GPIO_OE register controls the pin's output enable. When the OE bit is 1, the pin is driven LOW or HIGH according to the state of the corresponding OUT bit in the appropriate GPIO_OUT register. When the OE bit is 0, the output pin is tri-stated and can be used as an input.

Each IN bit in each GPIO_IN register contains the registered input value of a pin designated as a GPIO. The input goes through two D flip-flops to protect against meta-stability. The following figure shows how the pins are shared and multiplexed functionally.

Figure 23 SFIO/GPIO Pin Multiplexing Architecture



The SFIO output enable signals are generated automatically by the modules controlling the signals; they are not controlled through registers. However, the SFIO output enable mux does select which group of SFIO output enables signals to use based on the bits in the PIN_MUX_CTL registers.

These SFIO output enables are then sent to another mux that is controlled with the bits in the GPIO_CNF registers. Next, the selected output enables are ANDed with the bits in the TRI_STATE register. The ANDing function allows the output pins to be driven only if the TRI_STATE register bit is disabled.

In the input direction, the state of the input buffers is controlled with the DATAIN_EN signals. These signals are generated automatically by the modules controlling the signals; they are not controlled through registers. The first mux selects between the SFIO and GPIO input signals, and the last mux passes on the appropriate group of SFIO signals if so allowed.

14.2 GPIO 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 30 Tegra 3 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				
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				

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
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				
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				

Register Name	Offset (Lower: Read-Modify-Write)				Offset (Upper: Per-Pin Mask Write)			
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					

14.2.1 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: 000h..00fh | 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
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

Bit	Reset	Description
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

14.2.2 GPIO_OE_0

GPIO mode (GPIO_CNF.x=1) must be true for this condition to be valid.

Below set of registers 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: 010h..01fh | 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
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

Bit	Reset	Description
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

14.2.3 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 a 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: 020h..02fh | 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
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

Bit	Reset	Description
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

14.2.4 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: 030h..03fh | 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 programmed to generate an interrupt request. This configuration is also independent of that of the other pins.

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 the bit-3 of GPIO-PORT_C, write '1' into the bit-3 of

GPIO_INT.LVL.C register (this sets the interrupt to be active-on-high), and then write '1' into the 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. It has to be noted that the interrupt thus generated will be routed to the processor only if the corresponding bit for GPIO interrupts in the Secondary interrupt controller is enabled.

14.2.5 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 and Interrupt when switching from Low-High ot High-Low. Interrupt status for each port is saved in an Interrupt status registers.

This is an array of 4 identical register entries; the register fields below apply to each entry.

GPIO Port Interrupt Status Registers

Offset: 040h..04fh | 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

14.2.6 GPIO_INT_ENB_0

Every bit of GPIO pin needs has an enable which when enabled routes the Interrupt to 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: 050h..05h | 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

14.2.7 GPIO_INT_LVL_0

GPIO can detect interrupt for any edge /Trigger/Level signals. Following register bits [7:0] are set when Level signal toggles.

- Bits [15:8] toggles whenever an Edge changes High-Low.
- Bits [15:8] toggles whenever an Any Edge changes. High-Low or Low-High

This is an array of 4 identical register entries; the register fields below apply to each entry

GPIO Port Interrupt Activation Level Registers

Offset: 060h..06fh | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
23	0x0	DELTA_7: 1 means Trigger Interrupt on ANY change of input if EDGE is TRUE

Bit	Reset	Description
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
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

14.2.8 GPIO_INT_CLR_0

This register clears the Interrupts which are set. This is a Write-Only register. This is valid only in GPIO mode and 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: 070h..07fh | 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
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

14.2.9 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: 080h..08fh | 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
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

Bit	R/W	Reset	Description
1	RW	0x0	BIT_1: 0 = SPIO 1 = GPIO
0	RW	0x0	BIT_0: 0 = SPIO 1 = GPIO

14.2.10 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: 090h..09fh | 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
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

Bit	R/W	Reset	Description
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

14.2.11 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: 0a0h..0afh | 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
10	WO	0x0	MSK2: 0=Disable bit for write 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
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

14.2.12 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: 0c0h..0cfh | 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

14.2.13 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: 0d0h..0dfh | 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
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 = 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

Bit	R/W	Reset	Description
1	RW	0x0	BIT_1: 0 = DISABLE 1 = ENABLE
0	RW	0x0	BIT_0: 0 = DISABLE 1 = ENABLE

14.2.14 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: 0e0h..0efh | 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

Bit	R/W	Reset	Description
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

14.2.15 GPIO_LOCK_0

Lock bits are used to control the access to CNF and OE registers. When set, no one can write to CNF and OE bits. These can be programmed ONLY during Boot and gets 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: 0f0h..0ffh | 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

[THIS PAGE INTENTIONALLY LEFT BLANK]

15.0 CPU

The NVIDIA® Tegra® 3 CPU complex contains quad Cortex®-A9 CPUs and a PL310 L2 Cache controller. The L2 cache controller is described in another section under L2 Cache Controller.

The CPUs are quad ARM® Cortex-A9 MPCore™ processors, which is the SMP variant of the Cortex-A9 CPU (there is also a single core variant). These feature the NEON Media Processing Engine.

Tegra 3 devices implement the NVIDIA vSMP processor architecture, with 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. This provides efficient operation at low processor loads. These two CPU complexes operate exclusively to each other, meaning the hardware does not support simultaneous operation in any form. The L2 cache RAM is shared between the complexes allowing for fast transitions.

This section does not document the CPU itself; for that you will need the ARM documentation:

Cortex-A9 MPCore
Revision: r2p0
Technical Reference Manual

Published by ARM Limited, document number ARM DDI 0407D.

Cortex-A9 NEON Media Processing Engine
Revision: r2p0
Technical Reference Manual

Published by ARM Limited, document number ARM DDI 0409E.

Note:	Current version at this time. You should periodically look for updates and refer to the latest available version of this material.

Implementation Details

Many aspects of the Cortex-A9 processor may be configured for a specific implementation. The Tegra 3 implementation has the following features:

- r2p9 revision level
- 32 Kbyte I-cache per core
- 32 Kbyte D-cache per core
- 160 external interrupt inputs (SPIs)
- NEON Media Processing Engine, high performance vector floating point unit
- Trace supported (PTM)
- ACP present (but not used)

15.1 CPU Timers

The Cortex-A9 processor contains built in timers. These may be used by software, but are subject to some restrictions and warnings that the system timers external to the CPU complex are not.

These are:

- The timers are always clocked at the CPU clock speed divided by two. Therefore they will have to be re-programmed to maintain real time when the CPU clock frequency is changed.
- There is no hardware interaction between these timers and the flow controller, and the external interrupt unit, unlike the system timers. They are entirely internal to the Cortex-A9 processor.
- If the CPU is powered off, then they will stop counting and their contents will be lost unless saved and restored. They cannot be used to wake the CPU from a power down.

Refer to the Cortex-A9 documentation for further details of these timers.

16.0 LEVEL 2 CACHE CONTROLLER

The Tegra® 3 CPU complex contains quad ARM® Cortex®-A9 CPUs and a PL310 L2 Cache controller. This section describes the L2 cache controller. ARM also refers to this as the L2C-310 AMBA® Level 2 Cache Controller.

The L2 Cache Controller is an ARM PL310 L2 controller configured with 1 MByte of cache RAM.

This section does not document the PL310 itself; for that you will need the documentation that is available from ARM. At the time of writing the latest version is:

AMBA Level 2 Cache Controller (L2C-310)
Revision: r3p1
Technical Reference Manual

Published by ARM Limited, document number ARM DDI 0246E.

> **Note:** Current version at this time. You should periodically look for updates and refer to the latest available version of this material.

Implementation Details

Some aspects of the PL310 may be configured for a specific implementation. The Tegra 3 implementation has the following features:

- R3p1-50 revision level (RTL release 7)
- 8-way set-associative
- 1 Megabyte cache size
- Support for locking per master ID

[THIS PAGE INTENTIONALLY LEFT BLANK]

17.0 FLOW CONTROLLER

The flow controller provides a mechanism to halt processors 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 is halted in one of two ways:
  - Software Interaction: Software issues the WFE instruction to stop the processor, the flow controller asserts the EVENT input of the CPU to restart it.
  - Stop the clock [Legacy]: Should never be done for application software.
- Clocks to the processors can be halted when halted through flow control.

17.1 Flow Controller Registers

The EVENTS register is replicated five times, once each for COP (AVP) and each of CPU0-3

The fields and enums have the following interpretation:

- MODE defines how the flow controller logic operates; the reset value is 0x0 (does nothing).
- 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
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
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
FLOW_MODE_STOP_UNTIL_EVENT_AND_IRQ	6	Stop until event conditions met AND an interrupt controller interrupt occurs

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: These 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
- [IRQ,FIQ]_[1,0]: Resume on [IRQ,FIQ].[1,0] IRQ Valid, 1 is COP, 0 is CPU
- 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

uSEC, mSEC and SEC are based on free running clocks, so there is a -1 to 0 period possible difference between the programmed delay value and the really observed delay value.

17.1.1 FLOW_CTLR_HALT_CPU_EVENTS_0

Offset: 0x00 | 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

Bit	Reset	Description
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	IRQ_1
10	0x0	IRQ_0
9	0x0	FIQ_1
8	0x0	FIQ_0
7:0	0x0	ZERO

17.1.2 FLOW_CTLR_HALT_COP_EVENTS_0

Offset: 0x04 | 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

Bit	Reset	Description
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	IRQ_1
10	0x0	IRQ_0
9	0x0	FIQ_1
8	0x0	FIQ_0
7:0	0x0	ZERO

17.1.3 FLOW_CTLR_CPU_CSR_0

The CPU_CSR registers are replicated for each CPU core. 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 Power-Gate 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 the MPCore™
- PWR_OFF_STS: TRUE when CPU power-gated OFF by the flow controller
- INTR_FLAG: TRUE when Interrupt is Active -- Write-1-to-Clear
- EVENT_FLAG: TRUE when Event is Active -- Write-1-to Clear
- SWITCH_CLUSTER : If set, the active cluster is switched when all indicated CPUs reach STANDBY_WFI. This bit self-clears once the cluster switch sequence is completed
- WAIT_WFI_BITMAP : All cores indicated in the bitmap must be in STANDBY_WFI before switching Cluster
- WAIT_WFE_BITMAP: All cores indicated in bitmap must be in STANDBY_WFE before CPU Power-Gating
- EVENT_ENABLE: Generates an event when the flow controller exits the halted state
- ENABLE: Power-Gate Enable - Halt or Event-wait causes CPU Power-Gating

Offset: 0x08 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00xx000000000000)

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
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

17.1.4 FLOW_CTLR_COP_CSR_0

COP Control/Status for Interrupts

R/W addr=6000:700c

Offset: 0x0c | Read/Write: R/W | Reset: 0x00000000 (0b0)

Bit	Reset	Description
15	0x0	INTR_FLAG: TRUE when Interrupt is Active -- Write-1-to-Clear

17.1.5 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.

Bit	Reset	Description
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.

17.1.6 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
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	IRQ_1

Bit	Reset	Description
10	0x0	IRQ_0
9	0x0	FIQ_1
8	0x0	FIQ_0
7:0	0x0	ZERO

17.1.7 FLOW_CTLR_CPU1_CSR_0

Offset: 0x18 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx00xx000000000000)

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
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

17.1.8 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

Bit	Reset	Description
		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	IRQ_1
10	0X0	IRQ_0
9	0X0	FIQ_1
8	0X0	FIQ_0
7:0	0X0	ZERO

17.1.9 FLOW_CTLR_CPU2_CSR_0

Offset: 0x20 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00xx000000000000)

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

Bit	R/W	Reset	Description
16	RO	X	PWR_OFF_STS
15	RW	0x0	INTR_FLAG
14	RW	0x0	EVENT_FLAG
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

17.1.10 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
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

Bit	Reset	Description
14	0x0	OBF
13	0x0	IBE
12	0x0	IBF
11	0X0	IRQ_1
10	0X0	IRQ_0
9	0X0	FIQ_1
8	0X0	FIQ_0
7:0	0X0	ZERO

17.1.11 FLOW_CTLR_CPU3_CSR_0

Offset: 0x28 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx00xx000000000000)

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
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

17.1.12 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

Bit	Reset	Description
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 HW 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

[THIS PAGE INTENTIONALLY LEFT BLANK]

18.0 MEMORY CONTROLLER

The NVIDIA® Tegra® 3 memory controller is significantly enhanced from previous Tegra designs. Major features include:

- New arbiter design for higher memory efficiency.
- System Memory Management Unit (SMMU) for virtual to physical address mapping for any device.
- DDR3 and low voltage DDR3 support; as well as faster LP-DDR2 operation.

18.1 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.

18.2 Hardware Features

18.2.1 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-enable
  - 2 individually-controllable ODT (for DDR3)
- Each chip select may support different geometries
  - 2 or 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 up to 11 column bits[1], i.e. up to 4KB page-size
- Per-byte data masks
- BL4 or BL8 for both DDR3 and LPDDR2. For DDR3, BC4_on_the_fly.
- POP LPDDR2 up to 400 MHz; discrete LPDDR2 up to 533 MHz
- Discrete DDR3 up to 667 MHz
- 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

1 This includes C0, which is not pinned out on the DRAM, and C1, which will be zero for all burst-length four bursts.

- 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 3, we can set/adjust trims per-byte AND per-chip-select. We can still use the PVT-compensated value and either add a PVT/frequency-compensated adjustment to it (1/16, 1/8, 3/16 of a cycle) or a non-PVT- offset to it. There is no HW training mechanism. Either SW or characterization would need to determine best settings.
- Support for x32, x16, or x8 chips attached to the channel
  - DQ/DQS swizzling: MRR_BYTESEL need to match byte swizzling.

18.2.2 Memory Crossbar

The Memory Crossbar (MXB) is in charge of adapting Arbitration Domains to Protocol Arbiters (PAs). 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.

18.2.3 Global Resources

The Memory Subsystem also contains some global resources not owned by a particular AD or PA:

- SMMU
- Configuration registers and statistics counters

18.3 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

18.4 Initial Configuration

The details of configuring the Memory Subsystem (MSS) can be broken down into the following:

- Client configuration
- Protocol Arbiter configuration
- Arbitration Domain configuration
- Global Memory Subsystem configuration

> **Note:** This configuration must be performed as part of the chip-wide boot sequence

18.4.1 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 Mpixel per second = 1.72 µsec.

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 µsec ÷ 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.

18.4.2 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.

18.4.2.1 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.

18.4.2.2 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 has 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).

18.4.2.3 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

18.4.2.4 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.

18.4.3 Arbitration Domain Configuration

Physical implementation decisions (such as the mapping of clients to partition clients) combined with the requirement for unequal bandwidth allocation (eg. display clients needing 80% of the bandwidth) might necessitate the addition of configuration options to the Arbitration Domains to ensure the desired bandwidth allocation occurs.

18.4.4 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 ARM® 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 (aka “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. In general, the values of these should not need to be changed from the default value used by the MSS team, but 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.

[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.

- 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.

18.5 Power Management

The primary form of system power management involves 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 holdoff”*, which is a per-controller state bit. When the holdoff 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 holdoff is released.

The holdoff 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 holdoff is to group *all* requests for the DRAM into activity bursts.

The holdoff 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 holdoff from impacting these requests, software has a per-module configuration bit that disables the holdoff for certain known-latency-sensitive clients. Whenever a request with the holdoff-disable bit set enters the Protocol Arbiter, the holdoff 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.

18.6 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:

$$\text{Address} = \text{Surface\_base} + y * \text{Surface\_pitch} + x$$

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:

$$\text{Address} = \text{Surface\_base} + (y / 16) * (\text{Surface\_pitch} * 16) + (x / 16) * 256 + (y \,\%\, 16) * 16 + (x \,\%\, 16)$$

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 didn’t cause a bank-collision – to do this it needed to make sure that Surface_pitch wasn’t a multiple of 1 KB.

18.7 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.

18.8 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.
- B sends write request to flip the flag to “available”.
- B writes to the flag complete.
- A sees this and starts reading the packet data.
- B’s packet data writes complete.

Obviously in this scenario, 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 B can directly interrupt 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 3, 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 physical address are always column bits. Therefore, Tegra 3 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.

18.9 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.

18.10 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.

18.10.1 Boot

The Tegra 3 device has a two-stage boot:

- The Boot ROM is responsible for getting the chip out of reset and to the point 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 3 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 Tegra 3 core was powered off. Warm Boot use case occurs much more often; it is the use case hit whenever a SmartPhone wakes from Standby to answer a call.

18.10.1.1 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 MC/EMC based on BCT values

2. Wait for PLLs to lock
3. Program necessary always-on static pad configurations in PMC
4. Configure memory clocks based on BCT values
5. Enable MC/EMC clocks
6. Release memory subsystem resets
7. Program other necessary pad/pad-macro configuration
8. Program initial configuration for 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 200µs
    e. Deassert DRAM reset
    f. Apply stable clocks
    g. Wait 500µs
    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 periodical 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.

18.10.1.2 Warm Boot

The Warm Boot sequence is very similar to the Cold Boot sequence, and many of the same steps are performed. One obvious 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 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 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 Boot Loader 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.

[3] AHB_ARBITRATION_XBAR_CTRL.MEM_INIT_DONE

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 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 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 periodical 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.

18.10.2 Security Apertures

To secure the MC registers, the Cortex® A9 processor's page table entries for the MC register space must be marked as secured. Then to access the registers securely, the Cortex A9 must be switched into secure mode. For more details about these Cortex A9 operations, please read the Cortex A9 architecture manual from ARM.

To secure a region of EMEM, the Cortex A9'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 A9 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-A9 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 A9 in the secured mode:

```
RegWrite(MC_SECURITY_CFG2, carveout_base_address);
```

For more information on handling security violations, see subsection "MC Interrupts" below.

18.10.3 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 3, it is physically impossible to enable SMMU translation for the Cortex-A9 CPU.

There is no way to apply different ASIDs to requests that arrive at MC through the same MCCIF client. An example would be all three USB controllers access EMEM through the AHBSLV clients on Tegra 3; 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 subsection on “MC Interrupts” below.

18.10.3.1 SMMU Initialization

1. For each Virtual Address Space (VAS) to be used, set up the page-table in the DRAM. See 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 HW 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.

18.10.3.2 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

18.10.3.3 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.

18.10.4 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 suggests the following requirement:

- 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 HW Diagnostics feature that has power implications should be considered to be shadowed.
- No SW interference should be expected from SW after SW pulls the trigger to change frequency and before the clock change completes: SW runs on CPU from SDRAM. Once EMC blocks the requests to SDRAM, CPU may be frozen until the EMC finishes frequency change and unblocks the requests. Note that 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 results in a design as follows:

- CAR module handshakes with EMC on clock change. EMC handshakes with MC on shadow register latching. The whole process is done by hardware and transparent to SW.
- A rough estimation of clock change delay is 1-4us for lpddr2, 2-6us 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 the summary of the frequency change process.

- SW turns off features like self-refresh or auto-calibration if necessary to minimize their interference with DVFS sequence
- SW programs MC/EMC shadow registers for the new frequency
- SW programs EMC clock change flow control registers, such as entering self-refresh before clock change and programming SDRAM mode registers after clock change
- SW programs MC/EMC clock source register to trigger the clock change
- CAR sends clock change request to EMC (clock divider and/or clock source)
- EMC stalls all DRAM commands including refreshes
- EMC handshakes with 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
- EMC executes pre-clock change sequence such as entering self-refresh and programming mode register to turn off DLL
- EMC latches in shadowed registers and requests MC to latch in shadow registers
- EMC acknowledges CAR to change clock
- CAR changes MC/EMC clock
- CAR desserts the clock change request to EMC indicating the clock change is done

- EMC executes post-clock-change sequence, such as exiting self-refresh and programming SDRAM mode registers
- EMC desserts the clock change acknowledgement to CAR
- EMC unstalls dram commands including refresh and resumes incoming MC transactions
- SW waits 1us before continuing,

SW programming sequence on Tegra 3 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 5us 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 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 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 us for the actions above to complete
- Update shadow registers
  EMC_TIMING_CONTROL = 1
  Poll for latching completion: EMC_EMC_STATUS_TIMING_UPDATE_STALLED == 0

18.10.5 ZQ Calibration Sequence

ZQ calibration is a feature of LPDDR2 and DDR3 to calibrate Ron (the output driver) across PVT (process, voltage, temperature), and also ODT values on DDR3. EMC supports periodically sending ZQ calibration commands to both DRAMs, requiring only the initial configuration after booting. EMC also supports one time calibration commands at booting.

For cold booting, Boot ROM does ZQ-init on LPDDR2 and ZQ-long on DDR3, and then enables periodical ZQ-short.

For warm booting, Boot ROM does ZQ-long on both LPDDR2 and DDR3, and then enables periodical 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, though we don’t use it because SW 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, though we don’t use it because SW 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

Periodical calibration:

- EMC_ZCAL_INTERVAL: number of microseconds to wait between periodical ZQ commands. Program this register to 0 to disable periodical 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

18.10.6 MRR Sequence

Mode Register Read (MRR) is desirable on LPDDR2 for reading manufacturer ID to identify DRAM vendors, or reading back from 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

18.10.7 Auto-cal Sequence

Auto-cal is for periodically calibrating the output driver of Tegra 3 pads. EMC supports periodic auto-calibration without SW interference, usually initiated at booting.

To enable auto-calibration:

- Program AUTO_CAL_INTERVAL: the number of microseconds between two auto-calibration
- 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 micro seconds between each calibration steps

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

18.10.8 LP0/LP1 Entry/Exit

LP0 and LP1 are both power saving modes with DRAM in self-refresh mode. The only difference is that MC and 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 us for worst case DSR exit time
- Program EMC_REQ_CTRL to stall all transactions
  STALL_ALL_WRITES = 1
  STALL_ALL_READS = 1
- Wait until 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-ref; 2: dev1 is in self-ref; 3: both devices are in self-ref
- 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 “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 1us
  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 us
  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

18.10.9 Performance Tuning

The Tegra MSS is architected in such a way that, other than the per-client LA setting, only minimal performance tuning is required.

> **Note:** The capitalization in the below sections refers to the actual register name of the associated setting.

18.10.9.1 Arbitration Knobs

The suggested configuration values produced by the emc_reg_calc program are the best-known values, the SW 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 shmoo these values for best performance; the emc_reg_calc script has been used in the performance verification effort.

18.10.9.2 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-usecase basis for some clients.

18.10.9.3 Activity indication for DVFS

The Tegra MSS does not maintain statistics information for the 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.

18.10.9.4 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.

18.10.10 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 MC and 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 3; 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 the 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 of the UNMASKED interrupt vector bits are set, the MC will assert the interrupt to the central interrupt handling module.

18.10.10.1 MC Interrupts

The MC contains two classes of interrupt: 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 3, 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 records information about error, including:

- 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 (ie 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.

18.10.11 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” (aka SWGROUP or SWNAME). This is intended to allow 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 3 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-A9
MPCORELP	AXICIF_LP	No	Cortex-A9
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.
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.

18.11 Functionality

18.11.1 Addressing

The physical address space supported by 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 carveout 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 3, 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.

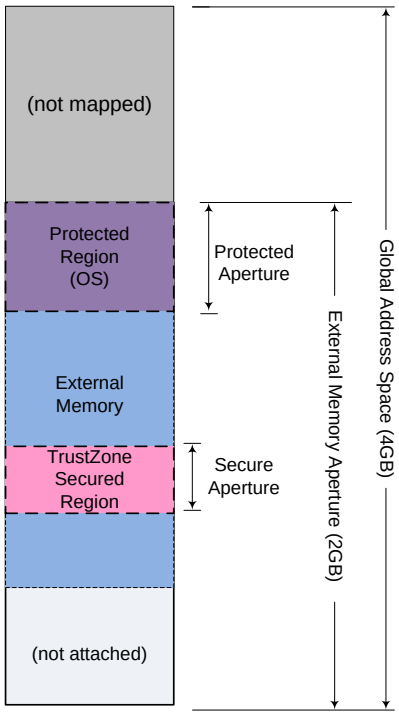
The MC receives a full 32 bit physical address and decodes it in the following way:

- 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.

[4] See subsection on TrustZone Security Region below for a discussion of security.

- A protected region of variable size can be defined in the external memory range, but always starts from offset 0x0.

Figure 24. Physical Address Map Example



Accesses that don't match one of the three address spaces are detected and an error logged. MC returns a predefined pattern[5] for reads to invalid addresses and acknowledges invalid write addresses, but otherwise EMC drops the write.

18.11.2 Granularity

The access granularity[6] inside 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 (aka 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.

18.11.3 Virtual Addressing

The SMMU is a centralized virtual-to-physical translation for MSS. It allows the following:

[5] All data bits are forced to 1's.

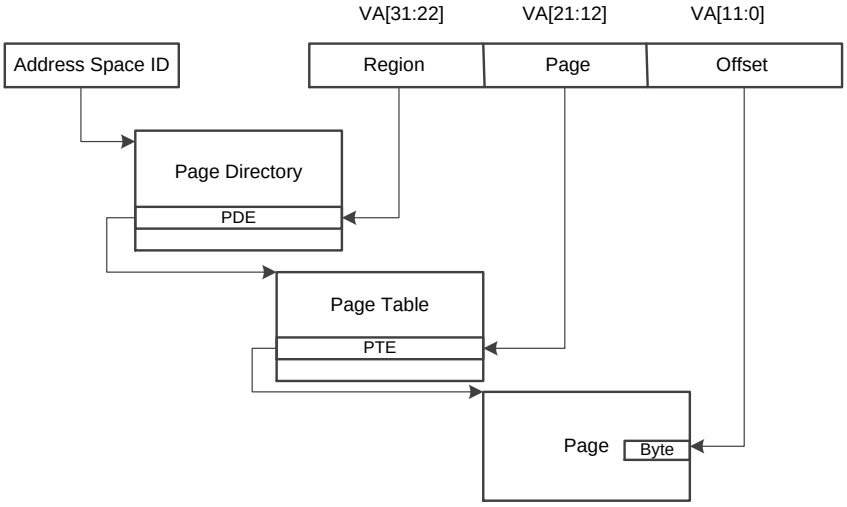
[6] A datum of 128-bit-granularity may be referred to as a "memory atom".

- 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 is via a two-level page table that carves up the virtual address as follows:

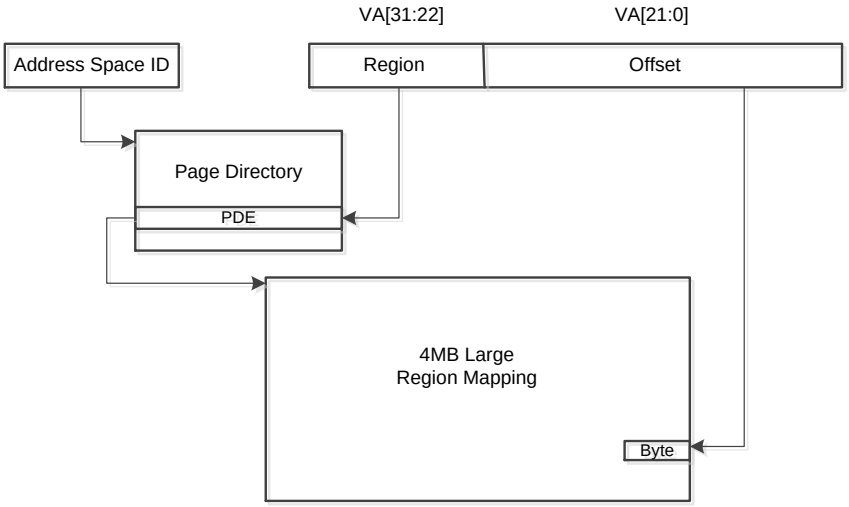
- Bits 31:22 choose a page directory entry (PDE). Each page directory has 1,024 PDEs, each mapping 4MB region.
- Bits 21:12 choose a page table entry (PTE) from a page table. Each page table has 1,024 PDEs, each mapping a 4kB page.
- Bits 11:0 choose a byte within a page

Figure 25. Page Table and Virtual Address



Each PTE or PDE consume 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 26. 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 provided by the implementation is a system parameter, but is expected to be a smallish number (Tegra 3 implements 4 ASIDs total). 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 HW 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 3 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).

18.11.3.1 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.

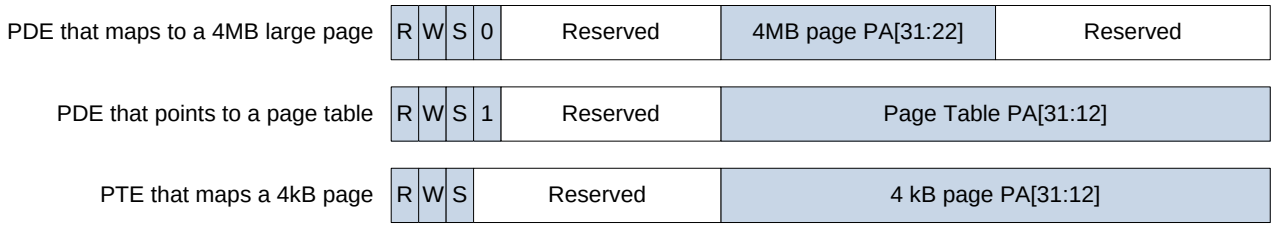
The in-memory layout of a PDE looks like:

- 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 27. 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.

18.11.3.2 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.

18.11.3.3 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.

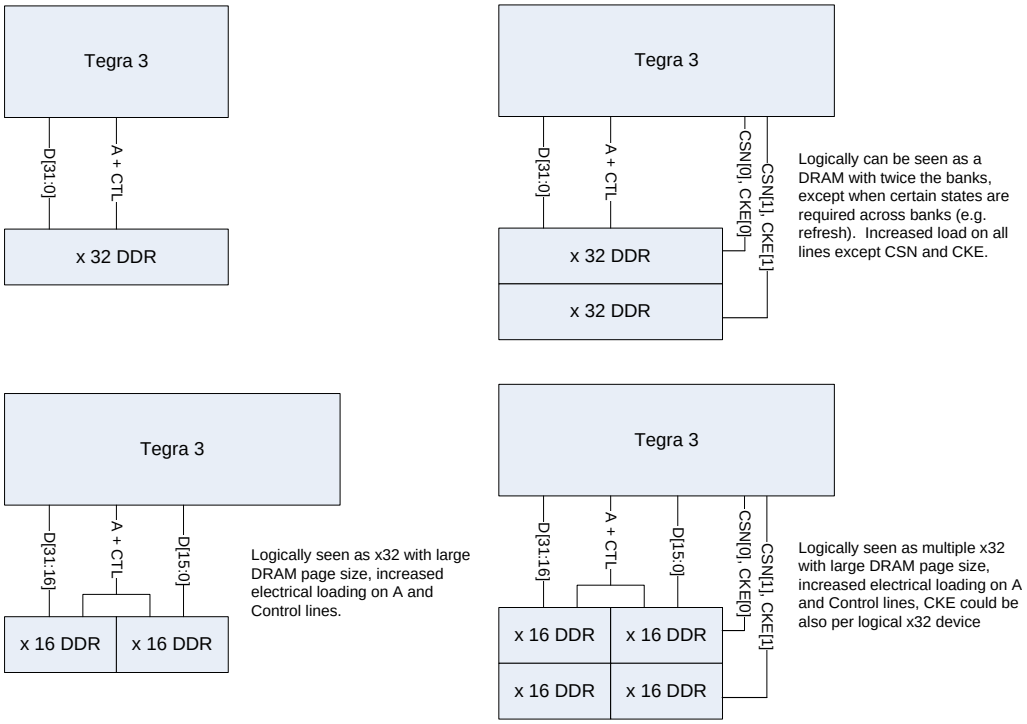
In addition to the TLB, page table updates require flushing stale entries out of the PTC. This needs to be done on a slot-by-slot basis by writing the PDE or PTE’s physical address (PA[31:4]) to the PTC flush command. Each flush flushes four aligned PTEs or PDEs (one DRAM atom’s worth) out of the cache.

Note that 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.

18.11.4 DRAM device, bank and page mapping

This relates to the external memory devices, EMC supports multiple devices with datawidths 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 28. 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.

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

[7] A x32 device may be a single x32 chip or two x16 chips in parallel.

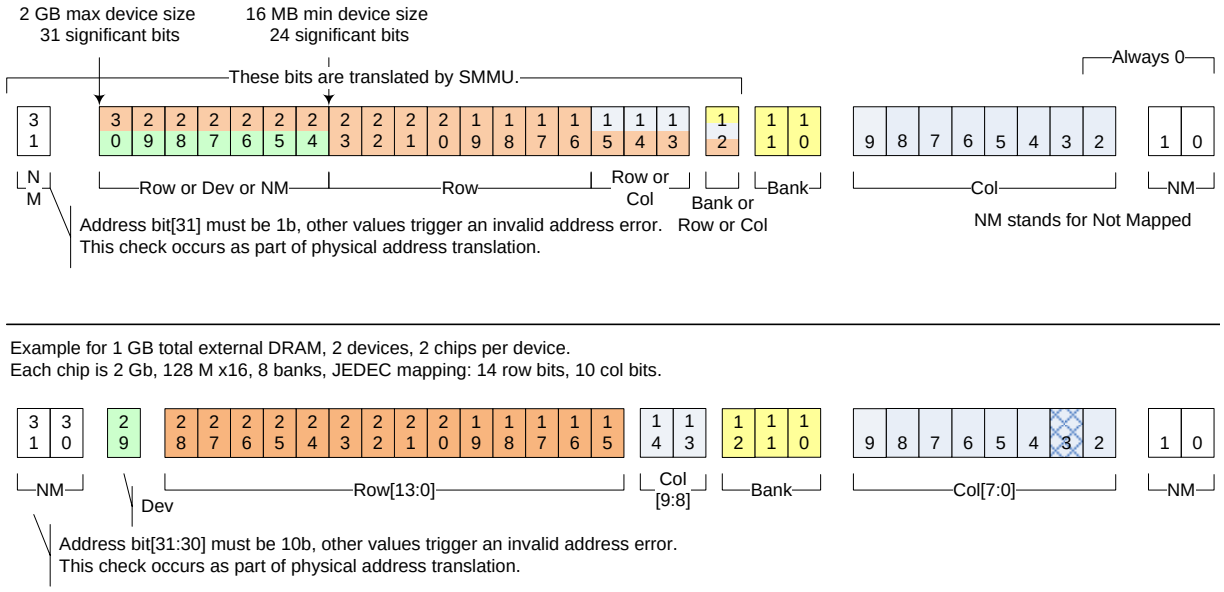
- Consecutive accesses to the same bank are for the same page in that bank

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 29. 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 you're using DDR3 with 1 device:
  - Yes: up to 16 bits
  - No: up to 15 bits
- Logical Devices (aka 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.

18.11.5 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 1MB. 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 3, 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.

18.11.6 OS-Protection Memory Region

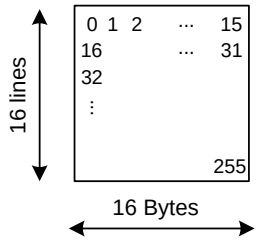
The MC can secure a region in memory from access and modification by the GPU hardware engines. The intended use of this section is to prevent modification of the OS image from untrusted software via the hardware engines. The protected region is defined to start at EMEM_BOM and can be extended in multiples 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.

18.12 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.

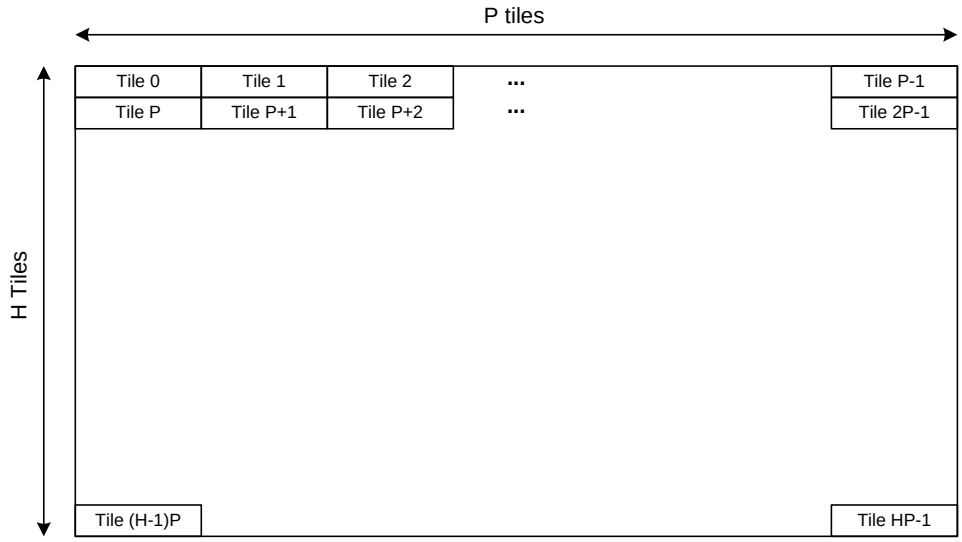
Figure 30. A Memory Tile



18.12.1 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 31. Tile Layout in a Surface



18.12.1.1 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.

18.12.1.2 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

$(linear – base) > 2^{22}$

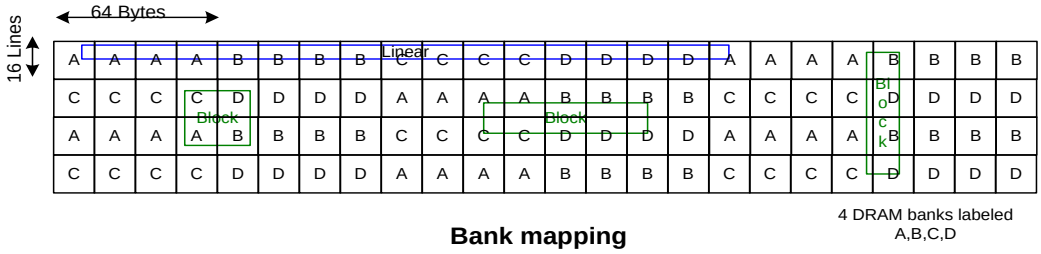
will result in incorrect tiled address. Certain surfaces that do not naturally align to a pitch of the form mentioned above will have wasted memory at the right and bottom edges of the surface.

18.12.2 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 3 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 32. Effects of Bank Address Mapping on Tile Layout



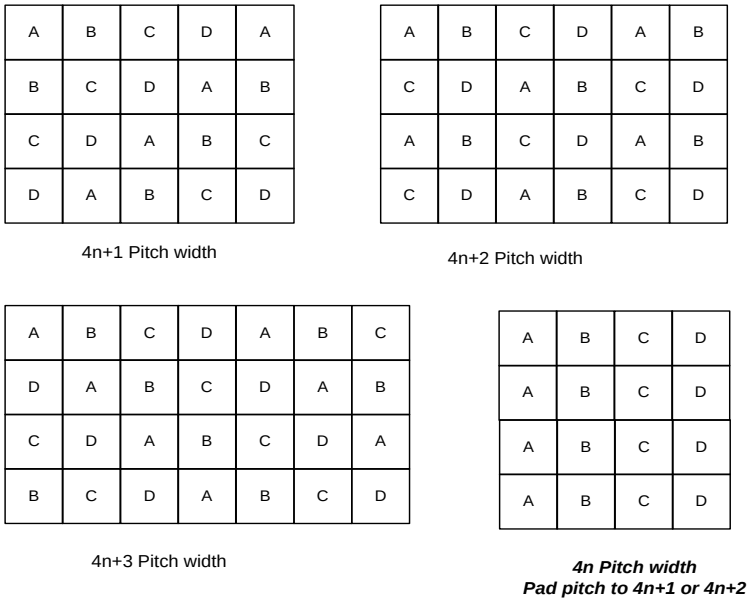
This bank mapping creates one problem, in that it reduces the inherent bank interleaving between different clients. The reordering aspects of the external memory arbiter should fairly handle contention between different clients; however, the basic resource conflict still limits the available performance of the system.

18.12.2.1 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 33. 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 (eg 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

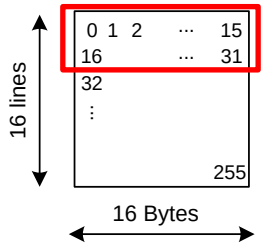
It is often the case that fetching from two surfaces which are tiled and switching buffers constantly. Good examples could be color and Z fetches, blending two surfaces, YUV surfaces, etc. In these cases it is quite possible that both the surfaces would keep accessing same bank and different rows, causing 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.

18.12.3 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 34. DDR3 Tiled Fetch



18.13 Memory Controller Registers

18.13.1 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 BootLoader as part of the warm boot (aka "wake from LP0") and cold boot (aka "power up") sequences. Suggested actions for the Boot ROM/Bootloader are noted in comments labeled "Boot requirements: ...".

USAGE NOTE: Some MC registers may only be accessed by TrustZone-secured register transactions. See the Cortex A9 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 bootrom code.

18.13.1.1 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: 010h | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0b0

Bit	Reset	Description
0	DISABLE	SMMU_ENABLE: 0 = DISABLE 1 = ENABLE

18.13.1.2 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: 014h | 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 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.

18.13.1.3 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: 018h | 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 regs 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

18.13.1.4 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: 01ch | Read/Write: R/W | Reset: 0bxx

Bit	Reset	Description
1:0	X	CURRENT_ASID: ASID used to address SMMU_PTB register

18.13.1.5 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: 020h | 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

18.13.1.6 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: 030h | 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

18.13.1.7 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: 034h | 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

18.13.1.8 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: 038h | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0b0000xxxxxxxxxxxx0000

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.

18.13.1.9 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: 050h | Read/Write: R/W | Reset: 0b011111111111

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.

18.13.1.10 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* regs 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: 054h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	N1	EMEM_NUMDEV: [PMC] number of populated DRAM devices. 0 = N1 1 = N2

18.13.1.11 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: 058h | Read/Write: R/W | Reset: 0b0100xxxxxx10xxxxx010

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

18.13.1.12 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: 05ch | Read/Write: R/W | Reset: 0b0100xxxxxx10xxxxx010

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

18.13.1.13 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	a9lp	Reads from Cortex-A9 Shadow CPU core via the L2 cache
csr_mpcorer	39	(0x27)	mpcore	mpcore	pi	Reads from Cortex-A9 4 CPU cores via the L2 cache
csw_mpcorelpw	56	(0x38)	mpcorelp	mpcorelp	a9lp	Writes from Cortex-A9 Shadow CPU core via the L2 cache
csw_mpcorew	57	(0x39)	mpcore	mpcore	pi	Writes from Cortex-A9 4 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: 070h | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0b100000000000

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.

18.13.1.14 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: 074h | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0b000000000000

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.

18.13.1.15 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	a9lp	Reads from Cortex-A9 Shadow CPU core via the L2 cache
csr_mpcorer	39	(0x27)	mpcore	mpcore	pi	Reads from Cortex-A9 4 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	a9lp	Writes from Cortex-A9 Shadow CPU core via the L2 cache
csw_mpcorew	57	(0x39)	mpcore	mpcore	pi	Writes from Cortex-A9 4 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: 078h | 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.

18.13.1.16 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: 090h | 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 (aka "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.

18.13.1.17 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 > max, 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: 094h | 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 holdoff 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.

18.13.1.18 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: 098h | Read/Write: R/W | Reset: 0b111111

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.

18.13.1.19 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: 09ch | Read/Write: R/W | Reset: 0b1111111

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. Note that: the combined value of RAP2PRE+RP 0x0 is illegal; the combined value of WAP2PRE+RP 0x0 is illegal.

18.13.1.20 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: 0a0h | Read/Write: R/W | Reset: 0b1111111

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.

18.13.1.21 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: 0a4h | Read/Write: R/W | Reset: 0b111111

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.

18.13.1.22 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: 0a8h | Read/Write: R/W | Reset: 0b111111

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.

18.13.1.23 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: 0ach | Read/Write: R/W | Reset: 0b1111

Bit	Reset	Description
3:0	0xf	RRD: The minimum number of cycles between a activate command and an activate command to a different bank. Program to ceil (EMC.RRD/DIV)-1.

18.13.1.24 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: 0b0h | Read/Write: R/W | Reset: 0b111111

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.

18.13.1.25 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: 0b4h | Read/Write: R/W | Reset: 0b1111111

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.

18.13.1.26 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: 0b8h | Read/Write: R/W | Reset: 0b11111

Bit	Reset	Description
4:0	0x1f	R2R: The number of cycles between consecutive read commands to different devices (aka, different chip-selects). Program to ceil (EMC.REXT/DIV)-1+OTFA+SFA.

18.13.1.27 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: 0bch | Read/Write: R/W | Reset: 0b11111

Bit	Reset	Description
4:0	0x1f	W2W: The number of cycles between consecutive write commands to different devices (aka, different chip-selects). Program to ceil (EMC.WEXT/DIV)-1+SFA.

18.13.1.28 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: 0c0h | Read/Write: R/W | Reset: 0b111111

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.

18.13.1.29 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: 0c4h | Read/Write: R/W | Reset: 0b111111

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.

18.13.1.30 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: 0d0h | Read/Write: R/W | Reset: 0b00001111000011110000001000000101

Bit	Reset	Description
31:24	0xf	W2R_TURN: Bubbles produced by a write to read bus turn. Approx MC_EMEM_ARB_TIMING_W2R.
23:16	0xf	R2W_TURN: Bubbles produced by a read to write bus turn. Approx MC_EMEM_ARB_TIMING_R2W.
15:8	0x2	W2W_TURN: Bubbles produced by a write to write (other device) bus turn. Approx MC_EMEM_ARB_TIMING_W2W.
7:0	0x5	R2R_TURN: Bubbles produced by a read to read (other device) bus turn. Approx MC_EMEM_ARB_TIMING_R2R.

18.13.1.31 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: 0d4h | Read/Write: R/W | Reset: 0b00001111000011110000011111

Bit	Reset	Description
23:16	0xf	RCD_W_COVER: Cost to cover the activate-to-write delay. Approx ceil ((EMC.RTP + EMC.RP + EMC.RD_RCD)/DIV)-1.
15:8	0xf	RCD_R_COVER: Cost to cover the activate-to-read delay. Approx 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. Approx MC_EMEM_ARB_TIMING_RC.

18.13.1.32 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 holdoff 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 holdoff.

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: 0d8h | Read/Write: R/W | Reset: 0b111001000001010x100000000010000

Bit	Reset	Description
30:28	0x7	ATOMS_PER_DVFS_PULSE: The MC will toggle the sys_stats_mon outputs for every

Bit	Reset	Description
		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, aka 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 (aka DIV=2). 1 = MC is the same frequency of EMC (aka DIV=1). 0 = DISABLE 1 = ENABLE
26:21	0x10	EXPIRING_SOON_SLACK_THRESHOLD: Slack threshold below which a 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.

18.13.1.33 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: 0dch | Read/Write: R/W | Reset: 0b10000x00

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

Bit	Reset	Description
		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

18.13.1.34 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: 0e0h | Read/Write: R/W | Reset: 0b11111xxxxxxxxxxx00000

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.

18.13.1.35 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: 0e4h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4:0	0x0	RING3_THROTTLE_CYCLES: Cycles of throttle after each request.

18.13.1.36 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: 200h | Read/Write: R/W | Reset: 0b000000000000000000

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
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

Bit	Reset	Description
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

18.13.1.37 MC_CLIENT_HOTRESET_STATUS_0

Contains one bit for each swname indicating the status of any flush that has been requested in the CLIET_HOTRESET_CTRL register.

NOTE: if no flush has been requested, register returns FLUSH_DONE.

Memory Client Hot Reset Status Register

Offset: 204h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
17	X	VI_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS
16	X	VDE_HOTRESET_STATUS: 1 = FLUSH_DONE

Bit	Reset	Description
		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
0	X	AFI_HOTRESET_STATUS: 1 = FLUSH_DONE 0 = FLUSH_IN_PROGRESS

18.13.1.38 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: 208h | Read/Write: R/W | Reset: 0b00000000000000011001000011111111

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
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

Bit	Reset	Description
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
0	0x1	ISO_PTCR: client ptcr is treated as an isochronous client 0 = DISABLE 1 = ENABLE

18.13.1.39 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: 20ch | 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
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

Bit	Reset	Description
		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
0	DISABLE	ISO_TEXSRD: client texsrd is treated as an isochronous client 0 = DISABLE 1 = ENABLE

18.13.1.40 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: 210h | Read/Write: R/W | Reset: 0b00

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

18.13.1.41 MC_EMEM_ARB_HYSTERESIS_0_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: 218h | Read/Write: R/W | Reset: 0b00000000000000011001000011111110

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
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

Bit	Reset	Description
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
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

Bit	Reset	Description
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

18.13.1.42 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: 21ch | 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
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

Bit	Reset	Description
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
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

Bit	Reset	Description
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

18.13.1.43 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: 220h | Read/Write: R/W | Reset: 0b00

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

18.13.1.44 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: 228h | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0b11111111111111111111111111111111

Bit	Reset	Description
31	ENABLE	SMMU_SATAR_ENABLE: enable client satar to be translated by SMMU 0 = DISABLE 1 = ENABLE
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

Bit	Reset	Description
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
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

Bit	Reset	Description
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

18.13.1.45 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: 22ch | 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
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

Bit	Reset	Description
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
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

Bit	Reset	Description
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

18.13.1.46 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: 230h | Read/Write: R/W | Secure: Trust Zone Protected | Reset: 0b11

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

18.13.1.47 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: 238h | 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

18.13.1.48 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: 23ch | 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

18.13.1.49 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: 240h | 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

18.13.1.50 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: 244h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

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

18.13.1.51 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: 248h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

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

18.13.1.52 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: 24ch | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

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

18.13.1.53 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: 250h | 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

18.13.1.54 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: 254h | 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

18.13.1.55 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: 258h | 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

18.13.1.56 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: 264h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

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

18.13.1.57 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: 268h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

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

18.13.1.58 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: 26ch | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

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

18.13.1.59 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: 270h | 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

18.13.1.60 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: 278h | 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

18.13.1.61 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: 27ch | 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

18.13.1.62 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: 280h | 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

18.13.1.63 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: 29ch | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_A9LP

18.13.1.64 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: 2a0h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_AHB

18.13.1.65 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: 2a4h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_APB

18.13.1.66 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: 2a8h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_AVP

18.13.1.67 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: 2ach | Read/Write: R/W | Reset: 0b11

Bit	Reset	Description
1:0	0x3	SNAP_LEVELS_DIS

18.13.1.68 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: 2b0h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_HEG

18.13.1.69 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: 2b4h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_ME

18.13.1.70 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: 2b8h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_PCX

18.13.1.71 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: 2bch | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_PI

18.13.1.72 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: 2c0h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_SAX

18.13.1.73 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: 2c4h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_TDA

18.13.1.74 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: 2c8h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_TDA2

18.13.1.75 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: 2cch | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_TDB

18.13.1.76 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: 2d0h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_TDB2

18.13.1.77 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: 2d4h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_VD

18.13.1.78 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: 2d8h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1:0	0x0	SNAP_LEVELS_VE

18.13.1.79 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: 2e0h | Read/Write: R/W | Reset: 0b00001100xxxxxxxx00010000

Bit	Reset	Description
23:16	0xc	ALLOWANCE_AFIW
7:0	0x10	ALLOWANCE_AFIR

18.13.1.80 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: 2e4h | Read/Write: R/W | Reset: 0b00001110xxxxxxxx00000100

Bit	Reset	Description
23:16	0xe	ALLOWANCE_AVPCARM7W
7:0	0x4	ALLOWANCE_AVPCARM7R

18.13.1.81 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: 2e8h | Read/Write: R/W | Reset: 0b01001110xxxxxxxx01001110

Bit	Reset	Description
23:16	0x4e	ALLOWANCE_DISPLAY0B
7:0	0x4e	ALLOWANCE_DISPLAY0A

18.13.1.82 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: 2ech | Read/Write: R/W | Reset: 0b01001110xxxxxxxx01001110

Bit	Reset	Description
23:16	0x4e	ALLOWANCE_DISPLAY1B
7:0	0x4e	ALLOWANCE_DISPLAY0C

18.13.1.83 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: 2f0h | Read/Write: R/W | Reset: 0b11111111

Bit	Reset	Description
7:0	0xff	ALLOWANCE_DISPLAYHC

18.13.1.84 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: 2f4h | Read/Write: R/W | Reset: 0b01001110xxxxxxxx01001110

Bit	Reset	Description
23:16	0x4e	ALLOWANCE_DISPLAY0BB
7:0	0x4e	ALLOWANCE_DISPLAY0AB

18.13.1.85 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: 2f8h | Read/Write: R/W | Reset: 0b01001110xxxxxxxx01001110

Bit	Reset	Description
23:16	0x4e	ALLOWANCE_DISPLAY1BB
7:0	0x4e	ALLOWANCE_DISPLAY0CB

18.13.1.86 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: 2fch | Read/Write: R/W | Reset: 0b11111111

Bit	Reset	Description
7:0	0xff	ALLOWANCE_DISPLAYHCB

18.13.1.87 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: 300h | Read/Write: R/W | Reset: 0b01101100xxxxxxxx00010111

Bit	Reset	Description
23:16	0x6c	ALLOWANCE_EPPU
7:0	0x17	ALLOWANCE_EPPUP

18.13.1.88 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: 304h | Read/Write: R/W | Reset: 0b01101100xxxxxxxx01101100

Bit	Reset	Description
23:16	0x6c	ALLOWANCE_EPPY
7:0	0x6c	ALLOWANCE_EPPV

18.13.1.89 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: 308h | Read/Write: R/W | Reset: 0b00001001xxxxxxxx00001001

Bit	Reset	Description
23:16	0x9	ALLOWANCE_G2SR
7:0	0x9	ALLOWANCE_G2PR

18.13.1.90 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: 30ch | Read/Write: R/W | Reset: 0b00001001xxxxxxxx00001010

Bit	Reset	Description
23:16	0x9	ALLOWANCE_G2DW
7:0	0xa	ALLOWANCE_G2DR

18.13.1.91 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: 310h | Read/Write: R/W | Reset: 0b01010000xxxxxxxx00000101

Bit	Reset	Description
23:16	0x50	ALLOWANCE_HOST1XR
7:0	0x5	ALLOWANCE_HOST1XDMAR

18.13.1.92 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: 314h | Read/Write: R/W | Reset: 0b00010000

Bit	Reset	Description
7:0	0x10	ALLOWANCE_HOST1XW

18.13.1.93 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: 318h | Read/Write: R/W | Reset: 0b11111111xxxxxxxx11111111

Bit	Reset	Description
23:16	0xff	ALLOWANCE_HDAW
7:0	0xff	ALLOWANCE_HDAR

18.13.1.94 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: 31ch | Read/Write: R/W | Reset: 0b11111111

Bit	Reset	Description
7:0	0xff	ALLOWANCE_ISPW

18.13.1.95 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: 320h | Read/Write: R/W | Reset: 0b00001110xxxxxxxx00000100

Bit	Reset	Description
23:16	0xe	ALLOWANCE_MPCOREW
7:0	0x4	ALLOWANCE_MPCORER

18.13.1.96 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: 324h | Read/Write: R/W | Reset: 0b00001110xxxxxxxx00000100

Bit	Reset	Description
23:16	0xe	ALLOWANCE_MPCORELPW
7:0	0x4	ALLOWANCE_MPCORELPR

18.13.1.97 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: 328h | Read/Write: R/W | Reset: 0b10000000xxxxxxxx01010000

Bit	Reset	Description
23:16	0x80	ALLOWANCE_MPE_IPRED
7:0	0x50	ALLOWANCE_MPEUNIFBR

18.13.1.98 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: 32ch | Read/Write: R/W | Reset: 0b11111111xxxxxxxx01000010

Bit	Reset	Description
23:16	0xff	ALLOWANCE_MPECSRD
7:0	0x42	ALLOWANCE_MPEAMEMRD

18.13.1.99 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: 330h | Read/Write: R/W | Reset: 0b11111111xxxxxxxx00010011

Bit	Reset	Description
23:16	0xff	ALLOWANCE_MPECSWR
7:0	0x13	ALLOWANCE_MPEUNIFBW

18.13.1.100 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: 334h | Read/Write: R/W | Reset: 0b00010011xxxxxxxx00001010

Bit	Reset	Description
23:16	0x13	ALLOWANCE_IDXSRD
7:0	0xa	ALLOWANCE_FDCDRD

18.13.1.101 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: 338h | Read/Write: R/W | Reset: 0b00001010xxxxxxxx00010011

Bit	Reset	Description
23:16	0xa	ALLOWANCE_FDCDWR
7:0	0x13	ALLOWANCE_TEXSRD

18.13.1.102 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: 33ch | Read/Write: R/W | Reset: 0b00010011xxxxxxxx00001010

Bit	Reset	Description
23:16	0x13	ALLOWANCE_IDXSRD2
7:0	0xa	ALLOWANCE_FDCDRD2

18.13.1.103 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: 340h | Read/Write: R/W | Reset: 0b00001010xxxxxxxx00010011

Bit	Reset	Description
23:16	0xa	ALLOWANCE_FDCDWR2
7:0	0x13	ALLOWANCE_TEXSRD2

18.13.1.104 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: 344h | Read/Write: R/W | Reset: 0b00010010xxxxxxxx00010000

Bit	Reset	Description
23:16	0x12	ALLOWANCE_PPCSAHBSLVR
7:0	0x10	ALLOWANCE_PPCSAHBDMAR

18.13.1.105 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: 348h | Read/Write: R/W | Reset: 0b00000110xxxxxxxx00010000

Bit	Reset	Description
23:16	0x6	ALLOWANCE_PPCSAHBSLVW
7:0	0x10	ALLOWANCE_PPCSAHBDMAW

18.13.1.106 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: 34ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	ALLOWANCE_PTCR

18.13.1.107 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: 350h | Read/Write: R/W | Reset: 0b00110011xxxxxxxx00110011

Bit	Reset	Description
23:16	0x33	ALLOWANCE_SATAW
7:0	0x33	ALLOWANCE_SATAR

18.13.1.108 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: 354h | Read/Write: R/W | Reset: 0b11010000xxxxxxxx11111111

Bit	Reset	Description
23:16	0xd0	ALLOWANCE_VDEMBER
7:0	0xff	ALLOWANCE_VDEBSEVR

18.13.1.109 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: 358h | Read/Write: R/W | Reset: 0b01110100xxxxxxxx00101010

Bit	Reset	Description
23:16	0x74	ALLOWANCE_VDETPER
7:0	0x2a	ALLOWANCE_VDEMCER

18.13.1.110 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: 35ch | Read/Write: R/W | Reset: 0b11111111xxxxxxxx11111111

Bit	Reset	Description
23:16	0xff	ALLOWANCE_VDEDBGW
7:0	0xff	ALLOWANCE_VDEBSEVW

18.13.1.111 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: 360h | Read/Write: R/W | Reset: 0b00101010xxxxxxxx01000010

Bit	Reset	Description
23:16	0x2a	ALLOWANCE_VDETPMW
7:0	0x42	ALLOWANCE_VDEMBEW

18.13.1.112 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: 364h | Read/Write: R/W | Reset: 0b00010010xxxxxxxx00101100

Bit	Reset	Description
23:16	0x12	ALLOWANCE_VIWSB
7:0	0x2c	ALLOWANCE_VIRUV

18.13.1.113 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: 368h | Read/Write: R/W | Reset: 0b10110010xxxxxxxx10110010

Bit	Reset	Description
23:16	0xb2	ALLOWANCE_VIWV
7:0	0xb2	ALLOWANCE_VIWU

18.13.1.114 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: 36ch | Read/Write: R/W | Reset: 0b00010010

Bit	Reset	Description
7:0	0x12	ALLOWANCE_VIWY

18.13.2 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 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 (aka "wake from LP0") and cold boot (aka "power up") sequences. Suggested actions for the Boot ROM/Boot Loader are noted in comments labelled "Boot requirements: ...".

USAGE NOTE: The EMC register fields to be stored in PMC must have "[PMC]", "[PMC2]", or "[PMC3]"... in their comments. ([PMC2] registers are packed/unpacked after [PMC] register group in SW code. [PMC3] is even later.) Internal note: After adding and removing such PMC flag, the tool warmboot_code_gen must be rerun to generate new boot ROM code.

18.13.2.1 EMC_INTSTATUS_0

Interrupt Status Register

Clear on 1-write. Init value is clear.

Offset: 000h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
7	CLEAR	DLL_ALARM_INT: indicate DLL alarm has been set 0 = CLEAR 1 = SET
6	CLEAR	ACCESS_TO_SR_DPD_DEV_INT: indicate 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

18.13.2.2 EMC_INTMASK_0

Interrupt Mask Register

Init value is masked.

Offset: 004h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
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

18.13.2.3 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: 008h | Read/Write: R/W | Reset: 0b1xxxxxxxxxx00x10xxxxx0000

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 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

Bit	Reset	Description
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 writing the TIMING_CONTROL 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

18.13.2.4 EMC_CFG_0

Configuration Register

The CFG register is used to configure the external memory interface.

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: 00ch | Read/Write: R/W | Reset: 0b0000xx11110

Bit	Reset	Description
31	DISABLED	DRAM_CLKSTOP_PD: clockstop only allowed to happen if DRAM is in active/precharge powerdown (for all CKE bits associated with clock) 0 = DISABLED 1 = ENABLED
30	DISABLED	DRAM_CLKSTOP_SR: clockstop only allowed to happen if DRAM is in self-refresh (for all CKE bits associated with clock) 0 = DISABLED 1 = ENABLED
29	NO_POWERDOWN	DRAM_ACPD: 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: 0 = DISABLED 1 = ENABLED
25	ENABLED	AUTO_PRE_WR: 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: 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

Bit	Reset	Description
		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: [PMC3] 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: [PMC3] 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. 0 = DISABLED 1 = ENABLED

18.13.2.5 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: 010h | Read/Write: R/W | Reset: 0b0xxxxxx0

Bit	Reset	Description
7	DISABLED	EMEM_ROW_MSB_ON_CS1: [PMC] drive CS1 pin with address bit 15 0 = DISABLED 1 = ENABLED
0	N1	EMEM_NUMDEV: number of populated DRAM devices. 0 = N1 1 = N2

18.13.2.6 EMC_REFCTRL_0

Refresh Control Register

The REFCTRL register allows SW 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 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: 020h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

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).

18.13.2.7 EMC_PIN_0

Controls state of selected DRAM pins

The PIN register allows SW 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: 024h | Read/Write: R/W | Reset: 0b1xxx0xxx0

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

18.13.2.8 EMC_TIMING_CONTROL_0

Triggers an update of the timing-related registers

The TIMING_CONTROL register is used by SW 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: 028h | Read/Write: R/W | Reset: 0bx

Bit	Reset	Description
0	X	TIMING_UPDATE

18.13.2.9 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: 02ch | Read/Write: R/W | Reset: 0b1111111

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

18.13.2.10 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: 030h | Read/Write: R/W | Reset: 0b000111111

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

18.13.2.11 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: 034h | Read/Write: R/W | Reset: 0b111111

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,

Bit	Reset	Description
		ceil(tRASmin/tCK) - (CMD_2T_TIMING == 1 ? 2 : 1)) LPDDR2: max(3, ceil(tRASmin/tCK)) - 1

18.13.2.12 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: 038h | Read/Write: R/W | Reset: 0b111111

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.

18.13.2.13 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.

Boot requirements

- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 03ch | Read/Write: R/W | Reset: 0b11111

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

18.13.2.14 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: 040h | Read/Write: R/W | Reset: 0b11111

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

18.13.2.15 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: 044h | Read/Write: R/W | Reset: 0b11111

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's 0

18.13.2.16 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: 048h | Read/Write: R/W | Reset: 0b111111

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)

18.13.2.17 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: 04ch | Read/Write: R/W | Reset: 0b011111

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

18.13.2.18 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: 050h | Read/Write: R/W | Reset: 0b011111

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

18.13.2.19 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: 054h | Read/Write: R/W | Reset: 0b1111

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

18.13.2.20 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: 058h | Read/Write: R/W | Reset: 0b0001

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

18.13.2.21 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: 05ch | Read/Write: R/W | Reset: 0b0000

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

18.13.2.22 EMC_QUSE_0

DRAM timing parameter

The QRST,QUSE,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

Because SDRAM uses a tristating clock (the DQS) a method is needed to deal with the ambiguity of when DQS is tristate. 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: 060h | Read/Write: R/W | Reset: 0b00010

Bit	Reset	Description
4:0	0x2	QUSE: [PMC] tells the chip when to look for read return data. LPDDR2/DDR3: obtained from characterization

18.13.2.23 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: 064h | Read/Write: R/W | Reset: 0b00001

Bit	Reset	Description
4:0	0x1	QRST: [PMC] time from expiration of QSAFE until reset is issued. DDR3 : CL - 2 LPDDR2 : RL - 2

18.13.2.24 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: 068h | Read/Write: R/W | Reset: 0b0111

Bit	Reset	Description
3:0	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

18.13.2.25 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 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.
- This register should be saved in the scratch registers and restored by the Boot ROM during warm boot.

Offset: 06ch | Read/Write: R/W | Reset: 0b01000

Bit	Reset	Description
4:0	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 150MHz where DRAM DLL has to be turned off 20 = MAX

18.13.2.26 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 133MHz, and the refresh requirement is 64ms 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: 070h | Read/Write: R/W | Reset: 0b0000000000011111

Bit	Reset	Description
15:6	0x0	REFRESH: [PMC] specifies the interval between refresh requests.
5:0	0x1f	REFRESH_LO

18.13.2.27 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: 074h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3:0	BR1	BURST_REFRESH_NUM: 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

18.13.2.28 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: 078h | Read/Write: R/W | Reset: 0b111110

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

18.13.2.29 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: 07ch | Read/Write: R/W | Reset: 0b111110

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

18.13.2.30 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: 080h | Read/Write: R/W | Reset: 0b011111

Bit	Reset	Description
5:0	0x1f	PCHG2PDEN: [PMC3] specify the timing delay from a precharge command to powerdown entry. DDR3 : 1 LPDDR2 : f_ceil(tRP]/tCK) - (CMD_2T_TIMING == 1 ? 1 : 0)

18.13.2.31 EMC_ACT2PDEN_0

[PMC3] 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: 084h | Read/Write: R/W | Reset: 0b011111

Bit	Reset	Description
5:0	0x1f	ACT2PDEN: specify the timing delay from an activate, mrs or emrs command to powerdown entry. DDR3/LPDDR2: 0

18.13.2.32 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: 088h | Read/Write: R/W | Reset: 0b000011111

Bit	Reset	Description
8:0	0x1f	AR2PDEN: [PMC3] specify the timing delay from an autorefresh command to powerdown entry. DDR3 : max(7, ceil((tXPDLL - tXP)/tCK)) LPDDR2 : 1

18.13.2.33 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: 08ch | Read/Write: R/W | Reset: 0b011111

Bit	Reset	Description
5:0	0x1f	RW2PDEN: [PMC3] 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

18.13.2.34 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: 090h | Read/Write: R/W | Reset: 0b1111111110

Bit	Reset	Description
9:0	0x3fe	TXSR: cycles between self-refresh exit & first DRAM command that doesn't require a locked DLL. Largest allowed value is 0x3fe DDR3: max(5, ceil(tXSR/tCK)) LPDDR2: max(2, ceil(tXSR/tCK))

18.13.2.35 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: 094h | Read/Write: R/W | Reset: 0b111110

Bit	Reset	Description
5:0	0x3e	TCKE: specify minimum CKE 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

18.13.2.36 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: 098h | Read/Write: R/W | Reset: 0b000000

Bit	Reset	Description
5:0	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)

18.13.2.37 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: 09ch | Read/Write: R/W | Reset: 0b000000

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

18.13.2.38 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: 0a0h | Read/Write: R/W | Reset: 0b01111

Bit	Reset	Description
4:0	0xf	TCLKSTABLE: 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

18.13.2.39 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: 0a4h | Read/Write: R/W | Reset: 0b01111

Bit	Reset	Description
4:0	0xf	TCLKSTOP: delay from last command to stopping the external clock to DRAM devices. DDR3 : max(5, ceil(tCKSRE/tCK)) - 1 LPDDR2

18.13.2.40 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: 0a8h | Read/Write: R/W | Reset: 0b00000000000000

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)

18.13.2.41 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 QUSE window earlier by 1 cycle, set QUSE_EXTRA = QUSE - 1. Disabled if QUSE_EXTRA = 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: 0ach | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3:0	0x0	QUSE_EXTRA: [PMC2] DDR3/LPDDR2: (EMC2TMC_CFG_XM2DQS_E_SCHMT == 1) ? (QUSE + 1) : 0

18.13.2.42 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: 0b0h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxxxxxx0x000

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.
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).

18.13.2.43 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: 0b4h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxxxxxx000

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.

18.13.2.44 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: 0b8h | Read/Write: R/W | Reset: 0b0000

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

18.13.2.45 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 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: 0bch | Read/Write: R/W | Reset: 0b01000

Bit	Reset	Description
4:0	0x8	CTT: [PMC] time from read command to CTT turn-on on DDR3 ? CTT_TERMINATION == 1 ? CL + 6 : 0 LPDDR2: 0 20 = MAX

18.13.2.46 EMC_MRS_WAIT_CNT_0

This register allows the delay between a MRS 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: 0c8h | Read/Write: R/W | Reset: 0b1000001000xxxxxx0000001111

Bit	Reset	Description
25:16	0x208	MRS_LONG_WAIT_CNT: Number of EMC clocks to wait before issuing any commands after sending MRS long command.
9:0	0xf	MRS_SHORT_WAIT_CNT: Number of EMC clocks to wait before issuing any commands after sending MRS short command.

18.13.2.47 EMC_MRS_0

Command trigger: MRS

The MRS register allows SW 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 HW 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: 0cch | Read/Write: R/W | Reset: 0b00xxx0xxxx00xxxxxx00000000000000

Bit	Reset	Description
31:30	0x0	MRS_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
26	SHORT	USE_MRS_LONG_CNT: Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	MRS_BA: Set to 0x0 for MRS.
13:0	0x0	MRS_ADR: mode-register data to be written.

18.13.2.48 EMC_EMRS_0

Command trigger: EMRS

The EMRS register allows SW 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: 0d0h | Read/Write: R/W | Reset: 0b00xxx0xxxx00xxxxxx00000000000000

Bit	Reset	Description
31:30	0x0	EMRS_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for only dev1.
26	SHORT	USE_EMRS_LONG_CNT: Indicate to use long or short MRS wait count. 0 = SHORT 1 = LONG
21:20	0x0	EMRS_BA: Set to 0x1 for EMRS (and where applicable, 0x2 for EMRS2, and 0x3 for EMRS3).
13:0	0x0	EMRS_ADR: mode-register data to be written.

18.13.2.49 EMC_REF_0

Command trigger: Refresh

The REF register allows SW to issue refresh commands. This is done to ensure proper DRAM initialization.

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: 0d4h | Read/Write: R/W | Reset: 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.

18.13.2.50 EMC_PRE_0

Command trigger: Precharge-All

The PRE register allows SW 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: 0d8h | Read/Write: R/W | Reset: 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.

18.13.2.51 EMC_NOP_0

Command trigger: NOP

The NOP register allows SW 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: 0dch | 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.

18.13.2.52 EMC_SELF_REF_0

Command trigger: SELF REFRESH

The SELF_REF register allows SW to issue self-refresh commands.

Do not program this register when a clock change sequence is on-going: check CLKCHANGE_COMPLETE_INT value if not sure.

Offset: 0e0h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

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.
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

18.13.2.53 EMC_DPD_0

Command trigger: Deep Power Down

The DPD register allows SW to issue a deep power down command.

Offset: 0e4h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

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

18.13.2.54 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: 0e8h | Read/Write: WO | Reset: 0b00xxxxxx00000000xxxxxxxx00000000

Bit	Reset	Description
31:30	0x0	MRW_DEV_SELECTN: active-low chip-select, 0x0 applies command to both devices, 0x2 to for only dev0, 0x1 for dev1.
23:16	0x0	MRW_MA: register address
7:0	0x0	MRW_OP: data to be written

18.13.2.55 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 is 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 ok to issue new MRR requests while there are on-going requests. e.g. to issue 3 MRR, follow these steps: 1,2,2,2,3,4,3,4,3,4. Data read back in step 4 are in the same order requested in step 2.

To make sure 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: 0ech | Read/Write: R/W | Reset: 0b00xxxxxx00000000xxxxxxxxxxxxxxxx

Bit	Reset	Description
31:30	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
23:16	0x0	MRR_MA: register address
15:0	X	MRR_DATA: data returned

18.13.2.56 EMC_XM2DQSPADCTRL3_0

Offset: 0f8h | Read/Write: R/W | Reset: 0b1000xxxxxxxxxxxxxxxxx0xxxxx

Bit	Reset	Description
27:24	0x8	EMC2PMACRO_CFG_XM2DQS_VREF_DQS: [PMC]
5	0x0	EMC2PMACRO_CFG_XM2DQS_E_VREF_DQS: [PMC]
0	X	EMC2PMACRO_CFG_XM2DQS_E_STRPULL_DQS: [PMC]

18.13.2.57 EMC_FBIO_CFG5_0

FBIO configuration Register

The FBIO_DQSIB_DLY registers have been removed.

The trimmer value may be overridden by setting using MULT==0 and OFFS = val <<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: 104h | Read/Write: R/W | Reset: 0b0000x0000xxx0001

Bit	Reset	Description
15:13	NORMAL	EMC2PMACRO_CFG_QUSE_MODE: [PMC] Select quse mode (NORMAL = off-chip) (ALWAYS_ON = requires DQS PULL) (INTERNAL_LPBK selects on-chip loopback) 0 = NORMAL 1 = ALWAYS_ON 2 = INTERNAL_LPBK 3 = PULSE_INT 4 = PULSE_EXT 5 = RESERVED
12	DISABLED	CMD_2T_TIMING: [PMC] enable 2T command timing (RAS/CAS/WE/ROW/A/BA). 0 = DISABLED 1 = ENABLED
10	DISABLED	DISABLE_CONCURRENT_AUTOPRE: 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 pulldowns on dqs lines (and pullups 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
7	DISABLED	DIFFERENTIAL_DQS: [PMC] enables differential signalling on dqs strobes (lpddr2/ddr2 options) 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

Bit	Reset	Description
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

18.13.2.58 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: 114h | Read/Write: R/W | Reset: 0b010

Bit	Reset	Description
2:0	0x2	CFG_QUSE_LATE: [PMC]

18.13.2.59 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: 2a4h | Read/Write: R/W | Reset: 0b0000xx001111x001xxx00000xxx00000

Bit	R/W	Reset	Description
31	RO	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	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: 1 (normal operation): use EMC generated pullup/dn (override or autocal) 0 (disabled): use emc2tmc_cfg* register settings for pullup/dn 0 = DISABLED 1 = ENABLED
28	RW	0x0	AUTOCAL_SLW_OVERRIDE: 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

Bit	R/W	Reset	Description
25:20	RW	0xf	AUTO_CAL_E_CAL_UPDATE: number of EMC clocks E_CAL_UPDATE is asserted
18:16	RW	0x1	AUTO_CAL_STEP: calibration step interval (in microseconds)
12:8	RW	0x0	AUTO_CAL_PD_OFFSET: [PMC] 2's complement offset for pull-down value
4:0	RW	0x0	AUTO_CAL_PU_OFFSET: [PMC] 2's complement offset for pull-up value

18.13.2.60 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: 2a8h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
20:0	0x0	AUTO_CAL_INTERVAL: 0: do calibration once Otherwise, auto-calibration occurs at intervals equivalent to the programmed number of microseconds.

18.13.2.61 EMC_AUTO_CAL_STATUS_0

EMC pad calibration status

Offset: 2ach | Read/Write: RO | Reset: 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: Pulldown code sent to pads
20:16	X	AUTO_CAL_PULLUP_ADJ: Pullup code sent to pads
12:8	X	AUTO_CAL_PULLDOWN: Pulldown code generated by auto-calibration
4:0	X	AUTO_CAL_PULLUP: Pullup code generated by auto-calibration

18.13.2.62 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: 2b0h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	STALL_ALL_WRITES: Stall incoming write transactions
0	0x0	STALL_ALL_READS: Stall incoming read transactions

18.13.2.63 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: 2b4h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23	X	TIMING_UPDATE_STALLED: indicate timing update triggered by TIMING_CONTROL.TIMING_UPDATE is not yet completed.
20	X	MRR_DIVLD: mrr data available for reading
19:16	X	MRR_FIFO_SPACE: mrr fifospace available
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

18.13.2.64 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 powerdown (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 setttings.

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: 2b8h | Read/Write: R/W | Reset: 0b00000000x0000000x00000000xx00011

Bit	Reset	Description
31	DISABLED	DRAMC_PRE_B4_ACT: 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

Bit	Reset	Description
22	0x0	ISSUE_PCHGALL_AFTER_REF: [PMC] Reserved bit. 1 = always generate precharge all bank after refresh.
21:20	0x0	MRR_BYTESEL_X16: [PMC3] If using 2 X16 DRAM on a single CS to form 32-bit wide data, indicates which byte lane 2nd DRAM's byte 0 is connected to.
19	0x0	BYPASS_POP_PIPE2_STAGE: For testing only [PMC] Reserved bit. 1 = bypass LPDDR2 adr/cmd pipeline stage2.
18	0x0	BYPASS_POP_PIPE1_STAGE: For testing only [PMC] Reserved bit. 1 = bypass LPDDR2 adr/cmd pipeline stage1.
17:16	0x0	MRR_BYTESEL: [PMC3] Indicates which AP byte lane is connected to DRAM byte 0 (over which MRR data is returned).
13	DISABLED	USE_PER_DEVICE_DLY_TRIM_OB: [PMC] Reserved bit. Allow using different outbound delay trim values per device. When enable, WDV must be program to >= 2, WEXT >=5. 0 = DISABLED 1 = ENABLED
12	DISABLED	USE_PER_DEVICE_DLY_TRIM_IB: [PMC] Reserved bit. Allow using different inbound delay trim values per device. When enable, REXT/RDV/QUSE must be program to >= 2. 0 = DISABLED 1 = ENABLED
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.
4	0x0	DIS_POP_PIPE2_STAGE_CLK: For testing only, must always be set to 1.
3	0x0	DIS_POP_PIPE1_STAGE_CLK: For testing only, must always be set to 1.
2	DISABLED	CLKCHANGE_SR_ENABLE: Forces DRAM into self-refresh during CLKCHANGE. Takes precedent over CLKCHANGE_PD_ENABLE if both are set. **Note**: 1. If the new clock change scheme is use, this bit must be disabled. 2. If this bit is enabled, DONT_USE_REF_REQACK_IFC must be set to 1. 3. If this big is enabled, DYN_SELF_REF must be set to DISABLED 0 = DISABLED 1 = ENABLED
1	ENABLED	CLKCHANGE_PD_ENABLE: Forces DRAM into power-down during CLKCHANGE. 0 = DISABLED 1 = ENABLED
0	ENABLED	CLKCHANGE_REQ_ENABLE: allows EMC and CAR to handshake on PLL divider/source changes. 0 = DISABLED 1 = ENABLED

18.13.2.65 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: 2bch | Read/Write: R/W | Reset: 0bx0000x00000000000000000010xx1x1

Bit	R/W	Reset	Description
31	RO	X	CFG_DLL_USE_OVERRIDE_UNTIL_LOCK: 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: Writing 1 to this register will send reset pulse to DLL's on next shadow update. Must reset DLL's 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 uSec. Counter is reset with DLL_RESET (from above) or with each periodic update (if using RUN_PERIODIC). Settings are: 00: LIMIT = 16uS 01: LIMIT = 64uS 10: LIMIT = 128uS 11: LIMIT = 512uS
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).
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: Enable DLL test mode
14	RW	0x0	CFG_DLL_QUSE_TESTOUT: Enable DLL TESTOUT on QUSE pads
13:12	RW	0x0	CFG_DLL_TESTSEL: Select DLL test output
11:8	RW	0x0	CFG_DLL_UDSET: [PMC] DLL Loop filter control (2^(udset+3)).
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 uSec to track delay changes (temp/voltage). 3 = RESERVED
5	RW	0x0	CFG_DLL_LOWSPEED: [PMC] Enable DLL for use with low-speed EMCCLK operation (<200MHz).
4	RO	X	CFG_DLL_STALL_RW_UNTIL_LOCK: [PMC] Writing 1 to this register will cause emc to stall all RW traffic until DLL locks. Takes effect on next shadow update.
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] Enable digital DLL 0 = DISABLED 1 = ENABLED

18.13.2.66 EMC_CFG_DIG_DLL_PERIOD_0

This register is shadowed: see usage note at top.

Offset: 2c0h | Read/Write: R/W | Reset: 0b1000000000000000

Bit	Reset	Description
15:0	0x8000	CFG_DLL_RUN_PERIOD: If CFG_DLL_MODE == RUN_PERIODIC, this specifies interval between runs in uSec

18.13.2.67 EMC_DIG_DLL_STATUS_0

Digital DLL Status

Digital DLL Status bits directly from DLL

Offset: 2c8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15	X	DLL_LOCK
14	X	DLL_ALARM
13	X	DLL_LOCK_TIMEOUT
9:0	X	DLL_OUT

18.13.2.68 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: 2d8h | Read/Write: R/W | Reset: 0b011

Bit	Reset	Description
2:0	0x3	CTT_DURATION: determines how long CTT remains enabled during reads. CTT determines when it will be enabled. DDR3: CTT_TERMINATION == 1 ? 3 : 0 LPDDR2: 0

18.13.2.69 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: 2dch | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxx01000xxxxx010

Bit	R/W	Reset	Description
31	RW	DISABLED	TERM_OVERRIDE: 0 = DISABLED 1 = ENABLED
28:24	RO	X	TERM_DRVUP
19:15	RO	X	TERM_DRVDN
12:8	RW	0x8	TERM_OFFSET:
2:0	RW	0x2	TERM_SLOPE:

ZQ Calibration Registers -- LPDDR2/DDR3

1. Periodic mode

Allows an 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 cmd to dev0, wait ZCAL_WAIT_CNT, send cmd to dev1, wait ZCAL_WAIT_CNT, resume normal operation. It will then wait ZCAL_INTERVAL microseconds before repeating the procedure. ZQ calibration command will not be sent to devices that are 1)unpopulated 2)either in or about to enter either 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 a MRW command.

Disable this feature by setting ZCAL_INTERVAL.ZCAL_REF_INTERVAL = 0, followed by TIMING_CONTROL to latch it.

2. Oneshot mode

Allows ZQ calibration command to be sent to dram.

To send a one-shot command, keep ZCAL_INTERVAL and ZCAL_WAIT_CNT 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 devices need 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 oneshot ZQ calibration command when periodic mode is enabled.

18.13.2.70 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: 2e0h | Read/Write: R/W | Reset: 0b00000000000000000000000000

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

18.13.2.71 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: 2e4h | Read/Write: R/W | Reset: 0b0000000000

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.

18.13.2.72 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: 2e8h | Read/Write: WO | Reset: 0b00xxxxxx00000000xxxxxxxx00000000

Bit	Reset	Description
31:30	0x0	ZQ_MRW_DEV_SELECTN: 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
7:0	0x0	ZQ_MRW_OP: [PMC] MRW OP field to be sent after ZCAL_INTERVAL

18.13.2.73 EMC_ZQ_CAL_0

Trigger a single ZQ Calibration

This register issues a ZQ calibration command for DDR3.

Offset: 2ech | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxxxxx1xxx0

Bit	Reset	Description
31:30	0x0	ZQ_CAL_DEV_SELECTN: active low chip-select, 0x0 applies command to both devices, 0x2

Bit	Reset	Description
		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).

18.13.2.74 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: 2f0h | Read/Write: R/W | Reset: 0b0010110xxxxx01000000

Bit	Reset	Description
22:16	0x16	EMC2PMACRO_CFG_XM2CMD_DIGTRIM: [PMC] DELAY trim setting for CMD CLKTRIM (skews cmds v. m2clk)
10	NORMAL	CFG_XM2CMD_CAL_SELECT: [PMC] Choose VREF/NORMAL (DQ/DQS) calibration for CMD pads. Use VREF for high freq LPDDR2_POP. 0 = NORMAL 1 = VREF
9	ENABLE	EMC2TMC_CFG_XM2RESET_E_DDR3: Select pad mode 0 = DISABLE 1 = ENABLE
8	DISABLE	EMC2PMACRO_CFG_XM2CMD_PHASESHIFT: [PMC2] shift cmd outputs by 1/2 cycle (useful for DDR3 in place of DIGTRIM) 0 = DISABLE 1 = ENABLE
7	DISABLE	EMC2TMC_CFG_XM2CMD_E_DDR3: Select pad mode 0 = DISABLE 1 = ENABLE
6	0x0	EMC2TMC_CFG_XM2CMD_CLK_SEL: [PMC2] pad clk_sel (ma bits get this value inverted in lpddr2 mode)
5	0x0	EMC2TMC_CFG_XM2CMD_E_PREEMP: [PMC] XM2CMD data pins preemp enable 0 = DISABLE 1 = ENABLE
4	0x0	EMC2TMC_CFG_XM2CMD_E_BYPASS: XM2CMD pins bypass outbound flop enable 0 = DISABLE 1 = ENABLE
3	0x0	EMC2TMC_CFG_XM2CMD_E_PWRD: XM2CMD pins pad powerdown signal 0 = DISABLE 1 = ENABLE

18.13.2.75 EMC_XM2CMDPADCTRL2_0

Offset: 2f4h | Read/Write: R/W | Reset: 0b000001010000110000 | Default: ffff858c.0000

Bit	Reset	SW Default	Description
31:28	0x0	0x8	EMC2PMACRO_CFG_XM2CMD_DRVUP_SLWF [PMC3]
27:24	0x5	0x5	EMC2PMACRO_CFG_XM2CMD_DRVDN_SLWR [PMC3]
23:19	0x1	0x11	EMC2PMACRO_CFG_XM2CMD_DRVUP [PMC3]
18:14	0x10	0x10	EMC2PMACRO_CFG_XM2CMD_DRVDN [PMC3]

18.13.2.76 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: 2f8h | Read/Write: R/W | Reset: 0b010010010000110000x10100xxx10100 | Default: ffff888c.1414

Bit	Reset	SW Default	Description
31:28	0x4	0x8	EMC2PMACRO_CFG_XM2DQS_DRVUP_SLWF [PMC3]
27:24	0x9	0x8	EMC2PMACRO_CFG_XM2DQS_DRVDN_SLWR [PMC3]
23:19	0x1	0x11	EMC2PMACRO_CFG_XM2DQS_DRVUP [PMC3]
18:14	0x10	0x10	EMC2PMACRO_CFG_XM2DQS_DRVDN [PMC3]
12:8	0x14	0x14	EMC2PMACRO_CFG_XM2DQS_DRVUP_TERM [PMC3]
4:0	0x14	0x14	EMC2PMACRO_CFG_XM2DQS_DRVDN_TERM [PMC3]

18.13.2.77 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: 2fch | Read/Write: R/W | Reset: 0b1000xxxxxxxxx011111000011000

Bit	Reset	Description
27:24	0x8	EMC2TMC_CFG_XM2DQS_VREF_DQ: [PMC]
14	0x0	EMC2TMC_CFG_XM2DQS_E_PWRD: XM2DQS pins pad powerdown signal 0 = DISABLE 1 = ENABLE
13	0x1	EMC2TMC_CFG_XM2DQS_E_SCHMT: [PMC] XM2DQS data pins schmidt 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

Bit	Reset	Description
		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: Select pad mode 0 = DISABLE 1 = ENABLE
7	0x0	EMC2TMC_CFG_XM2DQS_CLKSEL_DQS
6	0x0	EMC2TMC_CFG_XM2DQS_CLKSEL_DQ
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 preemp enable 0 = DISABLE 1 = ENABLE
1	0x0	EMC2TMC_CFG_XM2DQS_E_BYPASS: 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

18.13.2.78 EMC_XM2DQPADCTRL_0

CFG_XM2DQ Pad control register

These values only used if autocal is disabled

Offset: 300h | Read/Write: R/W | Reset: 0b01001001000011000001010011001 | Default: ffff888c.2990

Bit	Reset	SW Default	Description
31:28	0x4	0x8	EMC2PMACRO_CFG_XM2DQ_DRVUP_SLWF [PMC3]
27:24	0x9	0x8	EMC2PMACRO_CFG_XM2DQ_DRVDN_SLWR [PMC3]
23:19	0x1	0x11	EMC2PMACRO_CFG_XM2DQ_DRVUP [PMC3]
18:14	0x10	0x10	EMC2PMACRO_CFG_XM2DQ_DRVDN [PMC3]
13:9	0x14	0x14	EMC2PMACRO_CFG_XM2DQ_DRVUP_TERM [PMC3]
8:4	0x19	0x19	EMC2PMACRO_CFG_XM2DQ_DRVDN_TERM [PMC3]

18.13.2.79 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: 304h | Read/Write: R/W | Reset: 0b000x000x000x000

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

18.13.2.80 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: 308h | Read/Write: R/W | Reset: 0b11111111111111111110000000xx000x0

Bit	Reset	Description
31:28	0xf	EMC2PMACRO_CFG_XM2CLK_DRVUP_SLWF [PMC3]
27:24	0xf	EMC2PMACRO_CFG_XM2CLK_DRVDN_SLWR [PMC3]
23:19	0x1f	EMC2PMACRO_CFG_XM2CLK_DRVUP: [PMC] also used for the override of the second auto-cal cycle
18:14	0x1f	EMC2PMACRO_CFG_XM2CLK_DRVDN: [PMC] also used for the override of the second auto-cal cycle
13:11	0x0	EMC2TMC_CFG_XM2CLK_DLY_TRM_P [PMC3]
10:8	0x0	EMC2TMC_CFG_XM2CLK_DLY_TRM_N
7	DISABLE	EMC2TMC_CFG_XM2CLK_E_DDR3: Select pad mode 0 = DISABLE 1 = ENABLE
4	DISABLE	EMC2TMC_CFG_XM2CLK_E_PWRD: XM2CLK pins pad powerdown signal 0 = DISABLE 1 = ENABLE
3	DISABLE	EMC2TMC_CFG_XM2CLK_E_CAL_BYPASS: XM2CLK bypass drvdn/up calibration 0 = DISABLE 1 = ENABLE
2	DISABLE	EMC2TMC_CFG_XM2CLK_E_PREEMP: [PMC] preemp enable 0 = DISABLE 1 = ENABLE
0	DISABLE	EMC2PMACRO_CFG_DYN_PULLS_ON_CLKDIS: [PMC] Enable pullup on clkp and pulld on clkn when disabling clock. Reserved in case clock startup-ramp is too slow (DDR3). 0 = DISABLE 1 = ENABLE

18.13.2.81 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: 30ch | Read/Write: R/W | Reset: 0b11111xxx111110000000x1000

Bit	Reset	Description
24:20	0x1f	CFG_XM2COMP_DRVUP: used if AUTOCAL is disabled
16:12	0x1f	CFG_XM2COMP_DRVDN: used if AUTOCAL is disabled
11	DISABLE	EMC2TMC_CFG_XM2COMP_E_TESTOUT: 0 = DISABLE 1 = ENABLE
10	DISABLE	CFG_XM2COMP_VREF_CAL_EN: [PMC] When enable(=1), generate MCLK calibration cycle. 0 = DISABLE 1 = ENABLE
9	DISABLE	EMC2TMC_CFG_XM2COMP_E_PWRD: XM2CLK pins pad powerdown signal 0 = DISABLE 1 = ENABLE
8	DISABLE	EMC2TMC_CFG_XM2COMP_E_DDR3: Select pad mode 0 = DISABLE 1 = ENABLE
7:5	0x0	EMC2TMC_CFG_XM2COMP_BIAS_SEL: [PMC]
3:0	0x8	EMC2TMC_CFG_XM2COMP_VREF_SEL: [PMC]

18.13.2.82 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: 310h | Read/Write: R/W | Reset: 0b000xxxxx000x101x101x0x0x0x0

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]
6	0x0	EMC2TMC_CFG_XM2VTTGEN_CMD_E_PWRD: Select pad mode
4	0x0	EMC2TMC_CFG_XM2VTTGEN_DATA_E_PWRD: Select pad mode
2	0x0	EMC2TMC_CFG_XM2VTTGEN_E_DDR3: Select pad mode
0	0x0	EMC2TMC_CFG_XM2VTTGEN_SHORT: [PMC]

18.13.2.83 EMC_XM2VTTGENPADCTRL2_0

- E_TEST_BIAS: enables low current bias mode on VTTGEN cell.
- E_NO_VTTGEN: disables VTTGEN altogether. 1 VTTGEN pad in each MEM section is always enabled (DDR3-only)

Offset: 314h | Read/Write: R/W | Reset: 0b000000xxxxxxxxxxxxxxxxxxxxxxx111

Bit	Reset	Description
31:30	DISABLE	EMC2TMC_CFG_XM2VTTGEN_MEM2_E_TEST_BIAS: [PMC] Enable BIAS mode current mode during DPD -- MEM2 group 0 = DISABLE 1 = ENABLE 2 = RESERVED
29:28	DISABLE	EMC2TMC_CFG_XM2VTTGEN_MEM_E_TEST_BIAS: [PMC] Enable BIAS mode current mode during DP0 -- MEM group 0 = DISABLE 1 = ENABLE 2 = RESERVED
27:26	DISABLE	EMC2TMC_CFG_XM2VTTGEN_MEMCLK_E_TEST_BIAS: [PMC] Enable BIAS mode current mode during DPD -- MEM_CLK pad group 0 = DISABLE 1 = ENABLE 2 = RESERVED
2:0	0x7	EMC2TMC_CFG_XM2VTTGEN_E_NO_VTTGEN: [PMC] Disable optional VTTGEN pads (1 bit per pad) [0]--MEM_1, [1]--MEM_2, [0]--MEM2_1

18.13.2.84 EMC_XM2QUSEPADCTRL_0

XM2QUSE 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: 318h | Read/Write: R/W | Reset: 0b1000xxxxxxxxxxxxxx000101000

Bit	Reset	Description
27:24	0x8	EMC2TMC_CFG_XM2QUSE_VREF: [PMC]
8	DISABLE	EMC2TMC_CFG_XM2QUSE_E_DDR3: Select pad mode 0 = DISABLE 1 = ENABLE
7	0x0	EMC2TMC_CFG_XM2QUSE_CLK_SEL
6	DISABLE	EMC2TMC_CFG_XM2QUSE_E_PWRD: pad powerdown signal 0 = DISABLE 1 = ENABLE
5	ENABLE	EMC2TMC_CFG_XM2QUSE_E_SCHMT: [PMC] XM2DQ data pins schmidt enable 0 = DISABLE 1 = ENABLE
4	DISABLE	EMC2TMC_CFG_XM2QUSE_E_IVREF: [PMC] 0 = DISABLE 1 = ENABLE
3	ENABLE	EMC2TMC_CFG_XM2QUSE_E_CTT_HIZ: [PMC] 0 = DISABLE 1 = ENABLE
2	DISABLE	EMC2TMC_CFG_XM2QUSE_E_PREEMP: [PMC] CFG_XM2DQ data pins preemp enable 0 = DISABLE

Bit	Reset	Description
		1 = ENABLE
1	DISABLE	EMC2TMC_CFG_XM2QUSE_E_BYPASS: CFG_XM2DQ data pins bypass outbound flop enable 0 = DISABLE 1 = ENABLE
0	DISABLE	EMC2TMC_CFG_XM2QUSE_E_RX_FT_REC: [PMC] 0 = DISABLE 1 = ENABLE

18.13.2.85 EMC_EMCPADEN_0

EMC pad enable register

Writing 0 will disable listed direction

Offset: 31ch | Read/Write: R/W | Reset: 0b11

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

18.13.2.86 EMC_SCRATCH0_0

This is a scratch register for general use.

Offset: 324h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SCRATCH:

18.13.2.87 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. Default is multiply by 1, to keep DLL's 1/4 cycle delay. Integer + fractional formats for multiplier & offset:

OFFS is 2's complement format, with bit [4] representing integer

The output of the XFORM is xform_out = (xform_in * MULT + OFFS)/ 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 0x6000 < OFFS < (0x5fff - MULT * 0x3ff) DLL output is effectively overridden by setting XFORM_*_MULT to 0 and programming XFORM_*_OFFS[12:5] 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: 328h | Read/Write: R/W | Reset: 0b000000000000000xxx10000

Bit	Reset	Description
22:8	0x0	XFORM_DQS0_OFFS: [PMC]
4:0	0x10	XFORM_DQS0_MULT: [PMC]

18.13.2.88 EMC_DLL_XFORM_DQS1_0

Configure Digital DLL

Offset: 32ch | Read/Write: R/W | Reset: 0b000000000000000xxx10000

Bit	Reset	Description
22:8	0x0	XFORM_DQS1_OFFS: [PMC]
4:0	0x10	XFORM_DQS1_MULT

18.13.2.89 EMC_DLL_XFORM_DQS2_0

Configure Digital DLL

Offset: 330h | Read/Write: R/W | Reset: 0b000000000000000xxx10000

Bit	Reset	Description
22:8	0x0	XFORM_DQS2_OFFS: [PMC]
4:0	0x10	XFORM_DQS2_MULT

18.13.2.90 EMC_DLL_XFORM_DQS3_0

Configure Digital DLL

Offset: 334h | Read/Write: R/W | Reset: 0b000000000000000xxx10000

Bit	Reset	Description
22:8	0x0	XFORM_DQS3_OFFS: [PMC]
4:0	0x10	XFORM_DQS3_MULT

18.13.2.91 EMC_DLL_XFORM_DQS4_0

Configure Digital DLL

Offset: 338h | Read/Write: R/W | Reset: 0b000000000000000xxx10000

Bit	Reset	Description
22:8	0x0	XFORM_DQS4_OFFS: [PMC]

Bit	Reset	Description
4:0	0x10	XFORM_DQS4_MULT

18.13.2.92 EMC_DLL_XFORM_DQS5_0

Configure Digital DLL

Offset: 33ch | Read/Write: R/W | Reset: 0b0000000000000000xxx10000

Bit	Reset	Description
22:8	0x0	XFORM_DQS5_OFFS: [PMC]
4:0	0x10	XFORM_DQS5_MULT

18.13.2.93 EMC_DLL_XFORM_DQS6_0

Configure Digital DLL

Offset: 340h | Read/Write: R/W | Reset: 0b0000000000000000xxx10000

Bit	Reset	Description
22:8	0x0	XFORM_DQS6_OFFS: [PMC]
4:0	0x10	XFORM_DQS6_MULT

18.13.2.94 EMC_DLL_XFORM_DQS7_0

Configure Digital DLL

Offset: 344h | Read/Write: R/W | Reset: 0b0000000000000000xxx10000

Bit	Reset	Description
22:8	0x0	XFORM_DQS7_OFFS: [PMC]
4:0	0x10	XFORM_DQS7_MULT

18.13.2.95 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. Default is multiply by 1/2, to give 1/8 cycle delay.

OFFS is 2's complement format, with bit [4] representing integer. The output of the XFORM is xform_out = (xform_in * MULT + OFFS)/ 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 0x6000 < OFFS < (0x5fff - MULT * 0x3ff). DLL output is effectively overridden by setting XFORM_*_MULT to 0 and programming XFORM_*_OFFS[12:5] 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: 348h | Read/Write: R/W | Reset: 0b000000000000000xxx01000

Bit	Reset	Description
22:8	0x0	XFORM_QUSE0_OFFS: [PMC]
4:0	0x8	XFORM_QUSE0_MULT: [PMC]

18.13.2.96 EMC_DLL_XFORM_QUSE1_0

Configure Digital DLL

Offset: 34ch | Read/Write: R/W | Reset: 0b000000000000000xxx01000

Bit	Reset	Description
22:8	0x0	XFORM_QUSE1_OFFS: [PMC]
4:0	0x8	XFORM_QUSE1_MULT

18.13.2.97 EMC_DLL_XFORM_QUSE2_0

Configure Digital DLL

Offset: 350h | Read/Write: R/W | Reset: 0b000000000000000xxx01000

Bit	Reset	Description
22:8	0x0	XFORM_QUSE2_OFFS: [PMC]
4:0	0x8	XFORM_QUSE2_MULT

18.13.2.98 EMC_DLL_XFORM_QUSE3_0

Configure Digital DLL

Offset: 354h | Read/Write: R/W | Reset: 0b000000000000000xxx01000

Bit	Reset	Description
22:8	0x0	XFORM_QUSE3_OFFS: [PMC]
4:0	0x8	XFORM_QUSE3_MULT

18.13.2.99 EMC_DLL_XFORM_QUSE4_0

Configure Digital DLL

Offset: 358h | Read/Write: R/W | Reset: 0b000000000000000xxx01000

Bit	Reset	Description
22:8	0x0	XFORM_QUSE4_OFFS: [PMC]
4:0	0x8	XFORM_QUSE4_MULT

18.13.2.100 EMC_DLL_XFORM_QUSE5_0

Configure Digital DLL

Offset: 35ch | Read/Write: R/W | Reset: 0b000000000000000xxx01000

Bit	Reset	Description
22:8	0x0	XFORM_QUSE5_OFFS: [PMC]
4:0	0x8	XFORM_QUSE5_MULT

18.13.2.101 EMC_DLL_XFORM_QUSE6_0

Configure Digital DLL

Offset: 360h | Read/Write: R/W | Reset: 0b000000000000000xxx01000

Bit	Reset	Description
22:8	0x0	XFORM_QUSE6_OFFS: [PMC]
4:0	0x8	XFORM_QUSE6_MULT

18.13.2.102 EMC_DLL_XFORM_QUSE7_0

Configure Digital DLL

Offset: 364h | Read/Write: R/W | Reset: 0b000000000000000xxx01000

Bit	Reset	Description
22:8	0x0	XFORM_QUSE7_OFFS: [PMC]
4:0	0x8	XFORM_QUSE7_MULT

18.13.2.103 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 DQ_TRIMMER_RD register. Default is multiply by 1, providingg 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 [4] representing integer. The output of the XFORM is xform_out = (xform_in * MULT + OFFS)/ 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 0x6000 < OFFS < (0x5fff - MULT * 0x3ff)

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: 368h | Read/Write: R/W | Reset: 0b0000000000000000xxx10000

Bit	Reset	Description
22:8	0x0	XFORM_TXDQ0_OFFS: [PMC]
4:0	0x10	XFORM_TXDQ0_MULT: [PMC]

18.13.2.104 EMC_DLL_XFORM_DQ1_0

Configure Digital DLL

Offset: 36ch | Read/Write: R/W | Reset: 0b0000000000000000xxx10000

Bit	Reset	Description
22:8	0x0	XFORM_TXDQ1_OFFS: [PMC]
4:0	0x10	XFORM_TXDQ1_MULT

18.13.2.105 EMC_DLL_XFORM_DQ2_0

Configure Digital DLL

Offset: 370h | Read/Write: R/W | Reset: 0b0000000000000000xxx10000

Bit	Reset	Description
22:8	0x0	XFORM_TXDQ2_OFFS: [PMC]
4:0	0x10	XFORM_TXDQ2_MULT

18.13.2.106 EMC_DLL_XFORM_DQ3_0

Configure Digital DLL

Offset: 374h | Read/Write: R/W | Reset: 0b0000000000000000xxx10000

Bit	Reset	Description
22:8	0x0	XFORM_TXDQ3_OFFS: [PMC]
4:0	0x10	XFORM_TXDQ3_MULT

18.13.2.107 EMC_DLI_RX_TRIM0_0

Configure Digital DLL

Offset: 378h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_0
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_0

18.13.2.108 EMC_DLI_RX_TRIM1_0

Configure Digital DLL

Offset: 37ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_1
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_1

18.13.2.109 EMC_DLI_RX_TRIM2_0

Configure Digital DLL

Offset: 380h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_2
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_2

18.13.2.110 EMC_DLI_RX_TRIM3_0

Configure Digital DLL

Offset: 384h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_3
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_3

18.13.2.111 EMC_DLI_RX_TRIM4_0

Configure Digital DLL

Offset: 388h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_4
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_4

18.13.2.112 EMC_DLI_RX_TRIM5_0

Configure Digital DLL

Offset: 38ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_5
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_5

18.13.2.113 EMC_DLI_RX_TRIM6_0

Configure Digital DLL

Offset: 390h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_6
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_6

18.13.2.114 EMC_DLI_RX_TRIM7_0

Configure Digital DLL

Offset: 394h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
24:15	X	QUSE_CURRENT_TRIM_VAL_BYTE_7
9:0	X	DQS_CURRENT_TRIM_VAL_BYTE_7

18.13.2.115 EMC_DLI_TX_TRIM0_0

Configure Digital DLL

Offset: 398h | Read/Write: RO | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_0

18.13.2.116 EMC_DLI_TX_TRIM1_0

Configure Digital DLL

Offset: 39ch | Read/Write: RO | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_1

18.13.2.117 EMC_DLI_TX_TRIM2_0

Configure Digital DLL

Offset: 3a0h | Read/Write: RO | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_2

18.13.2.118 EMC_DLI_TX_TRIM3_0

Configure Digital DLL

Offset: 3a4h | Read/Write: RO | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	TXDQ_CURRENT_TRIM_VAL_BYTE_3

18.13.2.119 EMC_DLI_TRIM_TXDQS0_0

Set DLI TRIM

DLI_TRIM_TXDQSx

Controls trimmer used to skew databyte 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: 3a8h | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS0_DLI: [PMC]

18.13.2.120 EMC_DLI_TRIM_TXDQS1_0

Set DLI TRIM

Offset: 3ach | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS1_DLI: [PMC]

18.13.2.121 EMC_DLI_TRIM_TXDQS2_0

Set DLI TRIM

Offset: 3b0h | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS2_DLI: [PMC]

18.13.2.122 EMC_DLI_TRIM_TXDQS3_0

Set DLI TRIM

Offset: 3b4h | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS3_DLI: [PMC]

18.13.2.123 EMC_DLI_TRIM_TXDQS4_0

Set DLI TRIM

Offset: 3b8h | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS4_DLI: [PMC]

18.13.2.124 EMC_DLI_TRIM_TXDQS5_0

Set DLI TRIM

Offset: 3bch | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS5_DLI: [PMC]

18.13.2.125 EMC_DLI_TRIM_TXDQS6_0

Set DLI TRIM

Offset: 3c0h | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS6_DLI: [PMC]

18.13.2.126 EMC_DLI_TRIM_TXDQS7_0

Set DLI TRIM

Offset: 3c4h | Read/Write: R/W | Reset: 0b0000000

Bit	Reset	Description
6:0	0x0	EMC2PMACRO_CFG_TXDQS7_DLI: [PMC]

18.13.2.127 EMC_STALL_THEN_EXE_BEFORE_CLKCHANGE_0

Scheme to change controller timing parameters only

1. Program the EMC controller timing registers (shadowed)
2. Program CAR to change either clock source and/or clock divider

> **Note:** This scheme is not capable of changing read/write latency or enabling/disabling DLL. DONT_USE_REF_REQACK_IFC must be set to 1 in this scheme

Scheme to change controller timing parameters + SDRAM mode timing atomically.

1. Program the EMC controller timing registers (shadowed).
2. Pre-program SDRAM mode registers.
   a. Write 1 to STALL_THEN_EXE_BEFORE_CLKCHANGE field which will prevent further cfg execution until clock change request has been detected.

      Note: After this step, no register read should happen on EMC before the clock change (C) happens, otherwise system will hang. Disable EMC interrupt before this step if the interrupt handler reads EMC registers.

   b. 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,
   - Issue a write to SDRAM's MR1 register via EMC MRS register to disable DLL.
   - 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 (b); step (a) is still required.

   c. write 1 to STALL_THEN_EXE_AFTER_CLKCHANGE field which will prevent further cfg execution until after the actual clock change has happened.
   d. write to whatever EMC registers you want to be execute after clock change happened. Using the same example as (b), you will,
   - Issue a Self-refresh exit via EMC SELF_REF register.
   - Issue writes to SDRAM's MRx register(s) to change read/write latencies and/or enable DLL
   - Issue burst refreshs via EMC REF register.

      Note: If there's no need to execute any register access after clock change, simply skip step (d); step (c) is still required.

   e. write 1 to UNSTALL_RW_AFTER_CLKCHANGE field which will restore normal memory read/write accesses.
3. Program CAR to change either clock source and/or clock divider.

   Note: When using this clock change scheme, CLKCHANGE_SR_ENABLE field must be disabled.

   Note: In both sequences above, DYN_SELF_REF must be disabled off before the first step, then can be restored after the last step.

   The interval between two clock change sequences are recommended to be at least 20us apart

Write to this register will stall the register read/write path until a clock change request is detected. Once detected, whatever register access follows this register write will be executed until a STALL_THEN_EXE_AFTER_CKLCHANGE is encountered. At that point, EMC timing registers will be updated and clock change request will be acknowledge.

Offset: 3c8h | Read/Write: RO | Reset: 0b0

Bit	Reset	Description
0	0x0	STALL_THEN_EXE_BEFORE_CLKCHANGE: Writing a one to this bit will stall subsequent register access.

18.13.2.128 EMC_STALL_THEN_EXE_AFTER_CLKCHANGE_0

Write to this register will stall the register read/write path until after EMC timing registers are updated and that clock change has been completed. Once that happen, 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: 3cch | Read/Write: RO | Reset: 0b0

Bit	Reset	Description
0	0x0	STALL_THEN_EXE_AFTER_CLKCHANGE: Writing a one to this bit will stall subsequent register access.

18.13.2.129 EMC_UNSTALL_RW_AFTER_CLKCHANGE_0

Writing to this register will un-stall memory read/write after clock change, use in conjunction with STALL_THEN_EXE_AFTER_CLKCHANGE.

Offset: 3d0h | Read/Write: RO | Reset: 0b0

Bit	Reset	Description
0	0x0	UNSTALL_RW_AFTER_CLKCHANGE: Writing a one to this bit will unstall memory read/write after clock change.

18.13.2.130 EMC_AUTO_CAL_CLK_STATUS_0

EMC pad calibration status

Offset: 3d4h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:24	X	AUTO_CAL_CLK_PULLDOWN_ADJ: Pulldown code sent to pads for MCLK pad.
20:16	X	AUTO_CAL_CLK_PULLUP_ADJ: Pullup code sent to pads for MCLK pad.
12:8	X	AUTO_CAL_CLK_PULLDOWN: Pulldown code generated by auto-calibration for MCLK pad.
4:0	X	AUTO_CAL_CLK_PULLUP: Pullup code generated by auto-calibration for MCLK pad.

18.13.2.131 EMC_SEL_DPD_CTRL_0

Configures functional SEL_DPD modes

Offset: 3d8h | Read/Write: R/W | Reset: 0b100xxxxxx00xx000000

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
1	DISABLED	POP_CA_SEL_DPD_EN: [PMC] allow SEL_DPD assertion for pop command/address pads 0 = DISABLED 1 = ENABLED
0	DISABLED	POP_CLK_SEL_DPD_EN: [PMC] allow SEL_DPD assertion for pop clock pad 0 = DISABLED 1 = ENABLED

18.13.2.132 EMC_PRE_REFRESH_REQ_CNT_0

Offset: 3dch | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	PRE_REF_REQ_CNT: 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.

18.13.2.133 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 us 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 <= EABLED (re-enable DSR)
7. TIMING_UPDATE <= 1 (latch in the shadow values)

Offset: 3e0h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxx0000000000000000

Bit	Reset	Description
31	DISABLED	DSR_PER_DEVICE: controls whether dynamic self-refresh is done on a per-device basis. 0 = DISABLED 1 = ENABLED
15:0	0x0	DSR_THRESHOLD: number of idle cycles to wait before allowing dynamic self-refresh entry

18.13.2.134 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: 3e4h | Read/Write: R/W | Reset: 0b011111111111

Bit	Reset	Description
11:0	0x7ff	TXSRDLL: cycles between self-refresh exit & first DRAM command requiring a locked DLL DDR3: READ (and RAP) and synchronous ODT commands LPDDR2: NA Largest allowed value is 0xffe

[THIS PAGE INTENTIONALLY LEFT BLANK]

19.0 AHB

19.1 AHB Bus

The AHB Bus conforms to the ARM® *AMBA® Specification (Rev 2.0) Advanced High-performance Bus (AHB)* architecture as published by ARM Limited.

AHB is a 32-bit multi-master bus. Despite the nomenclature, it is considered a second tier bus in the Tegra® 3 device, 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 3 devices include:

- The AHB DMA controller master
- The AHB memory controller slave
- The AVP crossbar, both as a master and as a slave
- The APB DMA controller
- The USB-OTG controllers
- The NAND flash controller
- The NOR/MIO controller
- The SDMMC controllers
- The SE (security engine), both as a master and as a slave
- BSEA and BSEV bit-stream engines
- The CoreSight™ debug controller

AHB in Tegra 3 devices supports secure access to memory, using the same secure bit mechanism used by the ARM® Cortex®-A9 TrustZone™ security mechanism. This is only supported by the SE, which can make secure accesses to memory.

19.2 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 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 enable, 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 access.

Each AHB master is assigned to either the high or low priority bin.

19.2.1 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.

19.2.2 AHB Arbiter Registers

19.2.2.1 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: 004h | Read/Write: R/W | Reset: 0b0xxxxxxxxxx00000x000000x000x0000

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 2nd to last transfer. 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
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
16	0x0	BSEA: 1 = disable BSEA from arbitration. 0 = ENABLE 1 = DISABLE
14	0x0	SE: 1 = disable SE from arbitration. 0 = ENABLE 1 = DISABLE
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
3	0x0	CSITE: 1 = disable CoreSight from arbitration. 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
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

19.2.2.2 AHB_ARBITRATION_PRIORITY_CTRL_0

The AHB arbiter implements a 2-level priority scheme. In the 1st 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 2nd level, within each of the high/low priority group, arbitration is determined in a round-robin fashion.

AHB Arbitration Priority Control Register

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:29	0x0	AHB_PRIORITY_WEIGHT: AHB priority weight count. This 3-bit field is use 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.

19.2.2.3 AHB_ARBITRATION_USR_PROTECT_0

USR Protection Register

Offset: 00ch | Read/Write: R/W | Reset: 0b000000000

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
4	0x0	PPSB: Abort on USR mode access to PPSB memory space 0 = ABT_DIS 1 = ABT_EN
3	0x0	IRAMD: Abort on USR mode access to iRAMd memory space 0 = ABT_DIS

Bit	Reset	Description
		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

19.3 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.

19.3.1 AHB Gizmo Registers

19.3.1.1 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:** If the gizmo configuration bits are changed while the particular gizmo is transferring data, behavior of the system is non-deterministic.

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, 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's 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, gizmo will always generate split response for a read. If ' DON'T_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, 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, gizmo will break up the burst request internally into individual single word requests.
   - If 0, 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 gizmo will start write data request to the IP logic.
   - If 1, 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, 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, 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, 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. DON'T_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, 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, 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: 010h | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxx010xxx001

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 clk count 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
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 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 be think something is available in memory when it is in fact not. 0 = DISABLE 1 = ENABLE
7	0x1	DON'T_SPLIT_AHB_WR: AHB slave gizmo (memory controller) - don't split AHB write transaction 1 = don't 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's 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
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 = don't 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. 1 = enable 0 = disable. 0 = DISABLE 1 = ENABLE

19.3.1.2 AHB_GIZMO_APB_DMA_0

AHB Gizmo APB-DMA Control Register

Offset: 014h | Read/Write: R/W | Reset: 0b00000000xxxx1010

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 clk count 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

19.3.1.3 AHB_GIZMO_USB_0

AHB Gizmo USB Control Register

Offset: 020h | Read/Write: R/W | Reset: 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 clk count 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
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	DON'T_SPLIT_AHB_WR: AHB slave gizmo - don't split AHB write transaction. 1 = don't split AHB write transaction ever. 0 (and enable_split=1) = allow AHB write transaction to be split. 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
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's 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 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

19.3.1.4 AHB_GIZMO_AHB_XBAR_BRIDGE_0

AHB Gizmo AHB XBAR Bridge Control Register

Offset: 024h | Read/Write: R/W | Reset: 0b10001101

Bit	Reset	Description
7	0x1	DON'T_SPLIT_AHB_WR: AHB slave gizmo - don't split AHB write transaction. 1 = don't 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's 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

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	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 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

19.3.1.5 AHB_GIZMO_CPU_AHB_BRIDGE_0

AHB Gizmo CPU AHB Bridge Control Register

Offset: 028h | Read/Write: R/W | Reset: 0b00000000xxxx0110

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count 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

19.3.1.6 AHB_GIZMO_COP_AHB_BRIDGE_0

AHB Gizmo COP AHB Bridge Control Register

Offset: 02ch | Read/Write: R/W | Reset: 0b00000000xxxx0110

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count 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

19.3.1.7 AHB_GIZMO_XBAR_APB_CTLR_0

AHB Gizmo XBAR APB Control Register

Offset: 030h | Read/Write: R/W | Reset: 0b001

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

19.3.1.8 AHB_GIZMO_VCP_AHB_BRIDGE_0

AHB Gizmo VCP AHB Bridge Control Register

Offset: 034h | Read/Write: R/W | Reset: 0b00000000xxxx0110 |

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count 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

19.3.1.9 AHB_GIZMO_NAND_0

AHB Gizmo NAND Control Register

Offset: 040h | Read/Write: R/W | Reset: 0b00000000xxxx1010

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count 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

19.3.1.10 AHB_GIZMO_SDMMC4_0

This is now SDMMC4 master gizmo configuration

AHB Gizmo SDMMC4 Control Register

Offset: 048h | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx10xxx111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count 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 – Don't split AHB write transaction. 1 = don't 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's 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 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

19.3.1.11 AHB_GIZMO_SE_0

AHB Gizmo SE Control Register

Offset: 050h | Read/Write: R/W | Reset: 0b00000000xxxx0010

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.

Bit	Reset	Description
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

19.3.1.12 AHB_GIZMO_TZRAM_0

AHB Gizmo AHB TZRAM Control Register

Offset: 054h | Read/Write: R/W | Reset: 0b10xxx001

Bit	Reset	Description
7	0x1	DONT_SPLIT_AHB_WR: AHB slave gizmo – Don't split AHB write transaction. 1 = don't 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's 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 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

19.3.1.13 AHB_GIZMO_BSEV_0

This is now VDE's BSEV master gizmo configuration.

AHB Gizmo BSE Control Register

Offset: 064h | Read/Write: R/W | Reset: 0b00000000xxx00010

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.
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

19.3.1.14 AHB_GIZMO_BSEA_0

VDE's BSEA master gizmo configuration

AHB Gizmo BSEA Control Register

Offset: 074h | Read/Write: R/W | Reset: 0b00000000xxxxx010

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count 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

19.3.1.15 AHB_GIZMO_NOR_0

AHB Gizmo AHB NOR flash Control Register

Offset: 078h | Read/Write: R/W | Reset: 0b10xxx001

Bit	Reset	Description
7	0x1	DON'T_SPLIT_AHB_WR: AHB slave gizmo - don't split AHB write transaction. 1 = don't 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's 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 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

19.3.1.16 AHB_GIZMO_USB2_0

This is now USB2 master gizmo configuration

AHB Gizmo USB2 Control Register

Offset: 07ch | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx10xxx111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count between requests from this AHB master.

Bit	Reset	Description
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 – Don't split AHB write transaction. 1 = don't 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's 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 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

19.3.1.17 AHB_GIZMO_USB3_0

USB3 master gizmo configuration

AHB Gizmo USB3 Control Register

Offset: 080h | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx10xxx111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count 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

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 – Don't split AHB write transaction. 1 = don't 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's 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 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

19.3.1.18 AHB_GIZMO_SDMMC1_0

SDMMC1 master gizmo configuration

AHB Gizmo SDMMC1 Control Register

Offset: 084h | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx10xxx111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count 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 – Don't split AHB write transaction. 1 = don't split AHB write transaction ever. 0 (and enable_split=1) = allow AHB write transaction to be split.

Bit	Reset	Description
		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's 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 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

19.3.1.19 AHB_GIZMO_SDMMC2_0

SDMMC2 master gizmo configuration

AHB Gizmo SDMMC2 Control Register

Offset: 088h | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx10xxx111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count 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 – Don't split AHB write transaction. 1 = don't 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's 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

Bit	Reset	Description
		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 = don't force to single data transaction. 0 = NOT_SINGLE_DATA 1 = SINGLE_DATA
0	0x1	ENABLE_SPLIT: AHB slave gizmo - Enable splitting AHB transactions. 1 = enable 0 = disable 0 = DISABLE 1 = ENABLE

19.3.1.20 AHB_GIZMO_SDMMC3_0

SDMMC3 master gizmo configuration

AHB Gizmo SDMMC3 Control Register

Offset: 08ch | Read/Write: R/W | Reset: 0b00000000xxxxx010xxxxxxxx10xxx111

Bit	Reset	Description
31:24	0x0	REQ_NEG_CNT: AHB master request negate count. This is an 8-bit counter use to indicate the minimum number of clk count 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 - Don't split AHB write transaction. 1 = don't 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's 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 = don't 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. 1 = enable 0 = disable 0 = DISABLE; 1 = ENABLE

19.3.1.21 AHB_AHB_WRQ_EMPTY_0

Offset: 0c4h | Read/Write: RO | Reset: 0bxx

Bit	Reset	Description
1	X	COP_AHB_WRQ_EMPTY
0	X	CPU_AHB_WRQ_EMPTY

19.4 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.

19.4.1 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 four such pre-fetchers, allowing this function to be active for up to four 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.

19.4.1.1 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.

19.4.1.2 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 HW 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 HW 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.

19.4.2 AHB Memory Controller Slave Registers

19.4.2.1 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: 0dch | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3	0x0	DISABLE_CHECK_SIZE_MASTER4:
2	0x0	DISABLE_CHECK_SIZE_MASTER3:
1	0x0	DISABLE_CHECK_SIZE_MASTER2:
0	0x0	DISABLE_CHECK_SIZE_MASTER1:

19.4.2.2 AHB_ARBITRATION_XBAR_CTRL_0

XBAR Control Register

Offset: 0e0h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxx000

Bit	Reset	Description
17	0x0	SMMU_INIT_DONE: SW should set this bit when SMMU has been initialized 0 = NOT_DONE 1 = DONE
16	0x0	MEM_INIT_DONE: SW should set this bit when memory has been initialized

Bit	Reset	Description
		0 = NOT_DONE 1 = DONE
2	0x0	PPSB_STOPCLK_ENABLE: When TRUE enables sclk to be stopped for 1 cycle during ppsb read 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 SW handles coherency then this can be turned off to improve performance. SW writes to modify 0 = ENABLE 1 = DISABLE
0	0x0	POST_DIS: SW writes to modify 0 = ENABLE 1 = DISABLE

19.4.2.3 AHB_AHB_MEM_PREFETCH_CFG3_0

See the description of the pre-fetcher above.

Offset: 0e4h | Read/Write: R/W | Reset: 0b00010100100xxxxx0000100000000000

Bit	Reset	Description
31	0x0	ENABLE: 1=enable 0=disable
30:26	0x5	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 = 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+4) byte boundary. any value >16 will use n=16
15:0	0x800	INACTIVITY_TIMEOUT: 2048 cycles

19.4.2.4 AHB_AHB_MEM_PREFETCH_CFG4_0

See the description of the pre-fetcher above.

Offset: 0e8h | Read/Write: R/W | Reset: 0b00010100100xxxxx0000100000000000

Bit	Reset	Description
31	0x0	ENABLE: 1=enable 0=disable
30:26	0x5	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+4) byte boundary. any value >16 will use n=16
15:0	0x800	INACTIVITY_TIMEOUT: 2048 cycles

19.4.2.5 AHB_AVP_PPCS_RD_COH_STATUS_0

Offset: 0ech | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxx

Bit	Reset	Description
16	X	RDS_OUTSTANDING
0	X	WRS_OUTSTANDING

19.4.2.6 AHB_AHB_MEM_PREFETCH_CFG1_0

See the description of the pre-fetcher above.

Offset: 0f0h | Read/Write: R/W | Reset: 0b00010100100xxxxx0000100000000000

Bit	**Reset**	**Description**
31	0x0	ENABLE: 1=enable 0=disable

Bit	Reset	Description
30:26	0x5	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+4) byte boundary. any value >16 will use n=16
15:0	0x800	INACTIVITY_TIMEOUT: 2048 cycles

19.4.2.7 AHB_AHB_MEM_PREFETCH_CFG2_0

See the description of the pre-fetcher above.

Offset: 0f4h | Read/Write: R/W | Reset: 0b00011000100xxxxx0000100000000000

Bit	Reset	Description
31	0x0	ENABLE: 1=enable 0=disable

Bit	Reset	Description
30:26	0x6	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+4) byte boundary. any value >16 will use n=16
15:0	0x800	INACTIVITY_TIMEOUT:

19.4.2.8 AHB_AHBSLVMEM_STATUS_0

0x6000_C0F8: ahbslv outstanding rd, rdque_empty status

Offset: 0f8h | Read/Write: RO | Reset: 0bxx

Bit	Reset	Description
1	X	PPCS_RDS_OUTSTANDING
0	X	GIZMO_IP_RDQUE_EMPTY

19.4.2.9 AHB_ARBITRATION_AHB_MEM_WRQUE_MST_ID_0

AHB Memory Write Queue AHB Master ID Register

Offset: 0fch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
30:0	0x0	AHB_MASTER_ID: 0 = there is no write data in the write queue from that AHB master.

19.4.2.10 AHB_ARBITRATION_CPU_ABORT_ADDR_0

CPU Abort Address Register

Offset: 100h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR: Instruction Address which caused the abort.

19.4.2.11 AHB_ARBITRATION_CPU_ABORT_INFO_0

CPU Abort Info Register

Offset: 104h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

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
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

Bit	Reset	Description
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=hword, 10=word 0 = BYTE_ABT 1 = HWORD_ABT 2 = WORD_ABT

19.4.2.12 AHB_ARBITRATION_COP_ABORT_ADDR_0

CPU Abort Address Register

Offset: 108h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR: Instruction Address which caused the abort

19.4.2.13 AHB_ARBITRATION_COP_ABORT_INFO_0

COP Abort Info Register

Offset: 10ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

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

Bit	Reset	Description
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
1:0	X	SIZE: Aborted transaction Request Size 00=byte, 01=hword, 10=word 0 = BYTE_ABT 1 = HWORD_ABT 2 = WORD_ABT

19.4.2.14 AHB_AVPC_MCCIF_FIFOCTRL_0

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 2nd-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 Interface FIFO Control Register

Offset: 120h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxx0000

Bit	Reset	Description
17	0x0	AVPC_RCLK_OVERRIDE
16	0x0	AVPC_WCLK_OVERRIDE

Bit	Reset	Description
3	DISABLE	AVPC_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	AVPC_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	AVPC_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	AVPC_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

19.4.2.15 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: 124h | Read/Write: R/W | Reset: 0b00110010

Bit	Reset	Description
7:0	0x32	AVPCARM7W_WCOAL_TMVAL

19.4.2.16 AHB_MPCORELP_MCCIF_FIFOCTRL_0

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 2nd-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 Interface Fifo Control Register

Offset: 128h | Read/Write: R/W | Reset: 0b01

Bit	Reset	Description
17	0x0	SYS_REGS_MPCORELP_RCLK_OVERRIDE
16	0x1	SYS_REGS_MPCORELP_WCLK_OVERRIDE

19.4.2.17 AHB_MPCORE_MCCIF_FIFOCTRL_0

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 2nd-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 Interface Fifo Control Register

Offset: 12ch | Read/Write: R/W | Reset: 0b01

Bit	Reset	Description
17	0x0	SYS_REGS_MPCORE_RCLK_OVERRIDE
16	0x1	SYS_REGS_MPCORE_WCLK_OVERRIDE

19.4.2.18 AHB_AXICIF_FASTSYNC_CTRL_0

These registers control the Fast Synchronizer Logic in FIFOs between the CPU clock domain and the Memory Controller clock domain in the CPU AXICIF's MCCIF. This fast synchronizer will reduce latency, but is subject to certain restrictions stated below. These registers are nothing to do with the AHB bus, but are implemented here for convenience.

FASTSYNC_MODE

If ENABLE, the fastsync logic may be controlling the fifo's cross-clock synchronization. To enable this mode, the CLK_RST_CONTROLLER_LVL2_CLK_GATE_OVRC.MPCORE_MCCLK_OVR_ON must also be enabled to ensure 2nd-level clocks are always running.

When enabling fastsync:

- First set CLK_RST_CONTROLLER_LVL2_CLK_GATE_OVRC_0.MPCORE_MCCLK_OVR_ON == 1
- Then AHB_AXICIF_FASTSYNC_CTRL_0.FASTSYNC_MODE == 1

When disabling fastsync:

- first set AHB_AXICIF_FASTSYNC_CTRL_0.FASTSYNC_MODE == 0
- then if desired CLK_RST_CONTROLLER_LVL2_CLK_GATE_OVRC_0.MPCORE_MCCLK_OVR_ON == 0

While ENABLED, the following AXICIF_FASTSYNC(0|1|2)_* registers should not be written. The hardware will ignore writes to these registers while ENABLED.

To change the fastsync configuration:

- software must first set MODE=DISABLE
- readback MODE=DISABLE (to give it plenty of time to propagate)
- Update the configuration
- then set MODE=ENABLE.

FASTSYNC_EN_STATUS

Indication whether CAR is telling AXICIF it is safe to enable fast-mode. Note that these indicators lag the actual locking by a few cycles, so this should only be used to confirm that a software request to turn off fastsync has taken effect.

FASTSYNC_*CLK_TO_*CLK_STATUS

Indicates whether the fastsync logic is locked in fast-mode for the named clock-crossing. Note that these indicators lag the actual locking by a few cycles, so this should only be used to confirm that a software request to turn off fastsync has taken effect.

Offset: 130h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	R/W	Reset	Description
31	RO	X	FASTSYNC_MCCLK_TO_CPUCLK_STATUS: 0 = SLOW 1 = FAST
30	RO	X	FASTSYNC_CPUCLK_TO_MCCLK_STATUS: 0 = SLOW 1 = FAST
29	RO	X	FASTSYNC_EN_STATUS: 0 = DISABLE 0 = DISABLED 1 = ENABLE 1 = ENABLED
0	RW	DISABLE	FASTSYNC_MODE: 0 = DISABLE 1 = ENABLE

19.4.2.19 AHB_AXICIF_FASTSYNC_STATISTICS_0

Fastsync Performance Statistics

Allows the user to gather data about percentage of time fastsync logic is in fast synchronization mode. This statistics samples once a microsecond, (assuming that sys_clk period is greater than a microsecond).

FASTSYNC_STATS_GATHER –

The GATHER bits are master control bits to turn collection on and off. The stats controls can be enabled/disabled at will.

The RST state for GATHER zeros out any counters. Once entered, GATHER stays in the RST state until software changes it.

The DISABLE state causes all statistics counters to freeze at their current value. If the GATHER state goes back to ENABLE, the counters keep incrementing from their previous value.

The ENABLE state is when the statistics counters increment.

FASTSYNC_STATS_FAST_COUNT --

A count of microsecond samples where FASTSYNC_CPUCLK_TO_MCCLK_STATUS == FAST. If this counter hits its maximum value, both counters will stop counting and hold their value until cleared with GATHER=RST.

FASTSYNC_STATS_SLOW_COUNT --

A count of microsecond samples where FASTSYNC_CPUCLK_TO_MCCLK_STATUS == SLOW. If this counter hits its maximum value, both counters will stop counting and hold their value until cleared with GATHER=RST.

Note that these statistics use CPUCLK_TO_MCCLK because MCCLK is generally run slower than CPUCLK and thus there's more latency benefit when synchronization is fast in that direction.

Offset: 134h | Read/Write: R/W | Reset: 0b10xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31:30	RW	DISABLE	FASTSYNC_MCCLK_TO_CPUCLK_STATUS: 0 = SLOW 1 = FAST
29:15	RO	X	FASTSYNC_STATS_FAST_COUNT
14:0	RO	X	FASTSYNC_STATS_SLOW_COUNT

19.4.2.20 AHB_AXICIF_FASTSYNC0_CPUCLK_TO_MCCLK_0

Fastsync Per-direction controls:

ENABLE : If 1, allows fast synchronization (default: 0).

Note: can still get disabled temporarily while CAR is changing cpuclk or mcclk frequencies.

FREQ_FRAC_OVERRIDE: If 1, override frequency ratio fraction using FREQ_FREQ (default: 0, which means to use frequency estimator). Do not change value while FASTSYNC_CTRL.FASTSYNC_MODE==ENABLE.

PRED_INIT_FRAC: If not 0, overrides the up/lo bound initialization fraction used in the phase predictor (default: 0, which means compute fraction from max_delay parameter used for the phase detector). Do not change value while FASTSYNC_CTRL.FASTSYNC_MODE==ENABLE.

PRED_BOUND_FRAC: If not 0, overrides the up_bound comparison fraction used in the phase predictor (default: 0, which means compute fraction from min_delay parameter used for the phase detector). Do not change value while FASTSYNC_CTRL.FASTSYNC_MODE==ENABLE.

PLESIO_BOUND_FRAC: If not 0, overrides the plesio bound fraction used in the phase predictor when plesio mode is in effect; Note that this value is the upper 8 bits of the 17-bit fraction; (default: 0, which means to use a fraction which is 10% more than 2*max_delay param). Do not change value while FASTSYNC_CTRL.FASTSYNC_MODE==ENABLE.

PRED_INC_FRAC: If not 0, overrides the value subtracted-from/added-to low/high bounds each cycle until the next phase detect. It's essentially 2x the amount by which the prediction window widens each cycle before the next phase detect. (default: 0)

Do not change value while FASTSYNC_CTRL.FASTSYNC_MODE==ENABLE.

SPARE: 21 spare diagnostic workaround bits each direction (default: 0)

Do not change value while FASTSYNC_CTRL.FASTSYNC_MODE==ENABLE.

Offset: 138h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:21	0x0	SYS_REGS_FASTSYNC_CPUCLK_TO_MCCLK_SPARE0
20:4	0x0	SYS_REGS_FASTSYNC_CPUCLK_TO_MCCLK_FREQ_FRAC
3	0x0	SYS_REGS_FASTSYNC_CPUCLK_TO_MCCLK_FREQ_FRAC_OVERRIDE
2:1	0x0	SYS_REGS_FASTSYNC_CPUCLK_TO_MCCLK_SPARE
0	0x0	SYS_REGS_FASTSYNC_CPUCLK_TO_MCCLK_ENABLE

19.4.2.21 AHB_AXICIF_FASTSYNC1_CPUCLK_TO_MCCLK_0

Offset: 13ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	SYS_REGS_FASTSYNC_CPUCLK_TO_MCCLK_PRED_BOUND_FRAC
15:0	0x0	SYS_REGS_FASTSYNC_CPUCLK_TO_MCCLK_PRED_INIT_FRAC

19.4.2.22 AHB_AXICIF_FASTSYNC2_CPUCLK_TO_MCCLK_0

Offset: 140h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	SYS_REGS_FASTSYNC_CPUCLK_TO_MCCLK_PRED_INC_FRAC
15:8	0x0	SYS_REGS_FASTSYNC_CPUCLK_TO_MCCLK_SPARE2
7:0	0X0	SYS_REGS_FASTSYNC_CPUCLK_TO_MCCLK_PLESIO_BOUND_FRAC

19.4.2.23 AHB_AXICIF_FASTSYNC0_MCCLK_TO_CPUCLK_0

Offset: 144h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:21	0x0	SYS_REGS_FASTSYNC_MCCLK_TO_CPUCLK_SPARE0
20:4	0x0	SYS_REGS_FASTSYNC_MCCLK_TO_CPUCLK_FREQ_FRAC
3	0x0	SYS_REGS_FASTSYNC_MCCLK_TO_CPUCLK_FREQ_FRAC_OVERRIDE
2:1	0x0	SYS_REGS_FASTSYNC_MCCLK_TO_CPUCLK_SPARE
0	0x0	SYS_REGS_FASTSYNC_MCCLK_TO_CPUCLK_ENABLE

19.4.2.24 AHB_AXICIF_FASTSYNC1_MCCLK_TO_CPUCLK_0

Offset: 148h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	SYS_REGS_FASTSYNC_MCCLK_TO_CPUCLK_PRED_BOUND_FRAC
15:0	0x0	SYS_REGS_FASTSYNC_MCCLK_TO_CPUCLK_PRED_INIT_FRAC

19.4.2.25 AHB_AXICIF_FASTSYNC2_MCCLK_TO_CPUCLK_0

Offset: 14ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	SYS_REGS_FASTSYNC_MCCLK_TO_CPUCLK_PRED_INC_FRAC
15:8	0x0	SYS_REGS_FASTSYNC_MCCLK_TO_CPUCLK_SPARE2
7:0	0X0	SYS_REGS_FASTSYNC_MCCLK_TO_CPUCLK_PLESIO_BOUND_FRAC

19.5 AHB DMA Controller

The AHB DMA is a master placed between the AHB Bus and the Memory Controller (MC). It is used to accomplish block data transfers between external memory (DRAM) and internal memory (IRAM). It can also be used to move data from one section of external memory to another. It has four channels which can transfer data concurrently.

AHB DMA is used to transfer data from a source location to a destination location. Transfers can be from DRAM to DRAM, or between DRAM and IRAM in either direction. DMA transfers are done without any processor intervention other than the register writes to program the parameters for a particular transfer and handling interrupts.

AHB DMA is a “legacy” block, and is deprecated. Its performance is less than alternative memory to memory copy mechanisms such as program transfer (memcpy), or using the 2D engine. It may be useful as a means of DRAM to IRAM transfer but should not be used otherwise.

Features

- DMA master for transfers between external memory and AHB
- Data can be transferred from DRAM to DRAM, DRAM to IRAM or IRAM to DRAM
- Two modes of data transfer: single transfer (once) or continuous
- Programmable burst sizes of 1, 4, or 8 words
- Maximum transfer size of 64 KB per channel
- Minimum transfer size is 4B per channel
- Separate source and destination address pointers
- Per channel trigger and flow control mechanism
- Channel to channel trigger support i.e. ability to start transfer from one channel upon completion of transfer of another channel
- Interrupts per channel can be routed to CPU or AVP
- Ability to hold processor until transfer is done
- Address strides and wrap modes supported in once mode
- Double-buffering mode which allows transfer of data to two sequential destination addresses
- Round robin arbitration among channels at burst granularity
- Runs on system clock

19.5.1 Functionality

There are 4 channels in AHB DMA. An AHB DMA channel can transfer specified portions of data from an AHB address space (basically used for IRAM) to an MC address space, or transfer data between two different locations of MC address space.MC and XMB are used interchangeably. AHB DMA follows a simple round robin arbitration scheme.

Each channel can have independent burst transfer sizes programmed to one word, four words, or eight words.

Each DMA channel supports:

- Enable bit (note that there is another enable bit known as global enable bit).
- Direction bit to determine the direction of transfer: AHB to MC or MC to AHB.
- Interrupt Enable at the end of transfer with ability to mask or route to desired processor.
- Ability to hold off a Processor. Processor which writes into this bit will be held until transfer is completed.
- Trigger and Flow selects. These are additional control bits on which the transfer depends; Trigger bit is used to start a channel on some event to start the transfer and Flow bit is used to proceed with the transfer on every burst. This

again depends upon events from the system or AHB DMA itself. i.e., transfers can be started automatically, or under software control or under hardware control.

- Stride feature allows transfer of blocks of data where the transfer size and skip size are programmed.
- Wrap feature wraps the address on every burst transfer.
- Double Buffering Mode will make the AHB DMA Burst Address to reset to AHB Base Address after every even (2nd, 4th, 6th, etc) numbered transfer. Double buffering mode can be enabled with continuous transfer mode only.
- Separate registers for specifying AHB start address and MC start address.
- Ability to reduce the burst rate by delaying each burst by a programmable number of clock cycles.

19.5.2 Programming Guidelines

All the registers of a channel should be programmed before the channel enable bit is set.

Clearing the Global Enable bit causes the transfers that are in progress to be paused and setting the Global Enable resumes the transfers. If channel is disabled while transfer is in progress, transfer ends after completing any burst sequence that is in progress.

For best performance, MC source and destination addresses should be aligned to the burst size. For example, if Burst size is programmed to 4, the address needs to be aligned to 4-word boundary to achieve best throughput. If the MC address is not aligned to a 4-word boundary, 1-word transfers are initiated at the start and end of transfer as needed. Busy bit gets set as soon as a channel is enabled and gets cleared after transfer completes.

Interrupts are write-1-to-clear i.e., interrupt bit is cleared when the value of write data corresponding the bit position of the interrupt bit is 1.

19.5.3 AHB DMA Registers

19.5.3.1 AHBDMA_CMD_0

AHB-DMA Command Register

The Global Enable bit in the command register enables the AHB DMA. Clearing this bit causes active DMA transfers to be paused.

Pending bus transactions (ongoing burst) are completed and no new transactions are generated. Set this bit again to resume transfers.

Offset: 000h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
31	0x0	GEN: 0 = Disable AHB-DMA. 0 = DISABLE 1 = ENABLE

19.5.3.2 AHBDMA_STA_0

AHB-DMA Status Register

Busy bits in the status register indicate which (if any) of the AHB DMA channels have active pending AHB DMA transfers.

AHB DMA transfers that are started in continuous (repetitive) mode have their busy bits active until the enable bit for the AHB DMA channel is cleared to 0.

Offset: 004h | Read/Write: RO | Reset: 0bxxxx

Bit	Reset	Description
27	X	CH3: AHB DMA channel busy status flags set/cleared by HW 0 = NOT_BUSY 1 = BUSY
26	X	CH2: AHB DMA channel busy status flags set/cleared by HW 0 = NOT_BUSY 1 = BUSY
25	X	CH1: AHB DMA channel busy status flags set/cleared by HW 0 = NOT_BUSY 1 = BUSY
24	X	CH0: AHB DMA channel busy status flags set/cleared by HW 0 = NOT_BUSY 1 = BUSY

19.5.3.3 AHBDMA_TX_REQ_0

AHB-DMA Requestor Assignments

The Tx Requestors are values used as triggers or flow controls for AHB DMA transfers.

Offset: 008h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	SMP_31: Software requestor SMP31 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
30	X	SMP_30: Software requestor SMP30 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
29	X	SMP_29: Software requestor SMP29 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
28	X	SMP_28: Software requestor SMP28 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
27	X	AHB_3: End of AHB-DMA Transfer on Channel 3 0 = DISABLE 1 = ENABLE
26	X	AHB_2: End of AHB-DMA Transfer on Channel 2 0 = DISABLE 1 = ENABLE
25	X	AHB_1: End of AHB-DMA Transfer on Channel 1 0 = DISABLE 1 = ENABLE
24	X	AHB_0: End of AHB-DMA Transfer on Channel 0 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23	X	HRQ7_TMR2: Timer 2 Interrupt 0 = DISABLE 1 = ENABLE
22	X	HRQ6_TMR1: Timer 1 Interrupt 0 = DISABLE 1 = ENABLE
19	X	HRQ3_XRQ_D: 0 = NOP 0 = DISABLE 1 = ENABLE
18	X	HRQ2_XRQ_C: 0 = NOP 0 = DISABLE 1 = ENABLE
17	X	SMP_31_same_as_31_bit: Software requestor SMP31 from the SHRD_SMP.STA register (same as bit[31]) 0 = DISABLE 1 = ENABLE
16	X	SMP_30_same_as_30_bit: Software requestor SMP30 from the SHRD_SMP.STA register.(same as bit[30]) 0 = DISABLE 1 = ENABLE
15	X	SRQ1_XRQ_B: 0 = NOP 0 = DISABLE 1 = ENABLE
14	X	SRQ0_XRQ_A: 0 = NOP 0 = DISABLE 1 = ENABLE
11	X	Host1x: 0 = NOP 0 = DISABLE 1 = ENABLE
10	X	SMP_26: Enable software requestor SMP26 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
9	X	SMP_25: Enable software requestor SMP25 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
8	X	SMP_24: Enable software requestor SMP24 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
7	X	SMP_23: Enable software requestor SMP23 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
6	X	SMP_22: Enable software requestor SMP22 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5	X	SMP_21: Enable software requestor SMP21 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
4	X	SMP_20: Enable software requestor SMP20 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
3	X	SMP_19: Enable software requestor SMP19 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
2	X	SMP_18: Enable software requestor SMP18 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
1	X	SMP_17: Enable software requestor SMP17 from the SHRD_SMP.STA register. 0 = DISABLE 1 = ENABLE
0	X	CNTR_REQ: Enable Counter requestor. 0 = DISABLE 1 = ENABLE

19.5.3.4 AHBDMA_COUNTER_0

Controls which (if any) AHB DMA channel(s) will self-throttle based on the AHB DMA Counter Value. If any bit in [21:18] 4-bit field is active, then AHB DMA counter value will be enabled for decrement and reload (just as if CNTR.EN is set).

AHB-DMA Counter Usage

The AHB DMA Counter is used to slow down the request rates on some AHB DMA channels. It stores bits that configure which (if any) channel(s) should be throttled.

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
21	0x0	CH3_FL_CNT: Controls AHB DMA channel3 self-throttle based on the AHB DMA Counter Value. If this bit field is active, then AHB DMA counter value will be enabled for decrement and reload (just as if CNTR.EN is set). 0 = DISABLE 1 = ENABLE
20	0x0	CH2_FL_CNT: Controls AHB DMA channel2 self-throttle based on the AHB DMA Counter Value. If this bit field is active, then AHB DMA counter value will be enabled for decrement and reload (just as if CNTR.EN is set). 0 = DISABLE 1 = ENABLE
19	0x0	CH1_FL_CNT: Controls AHB DMA channel1 self-throttle based on the AHB DMA Counter Value. If this bit field is active, then AHB DMA counter value will be enabled for decrement and reload (just as if CNTR.EN is set). 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18	0x0	CH0_FL_CNT: Controls AHB DMA channel0 self-throttle based on the AHB DMA Counter Value. If this bit field is active, then AHB DMA counter value will be enabled for decrement and reload (just as if CNTR.EN is set). 0 = DISABLE 1 = ENABLE
17	0x0	CNTR_EN: When this bit is set Counter is enabled. 0 = DISABLE 1 = ENABLE
16	0x0	PRD_EN: Normally, AHB DMA current count value is reloaded to the programmed init/reload value whenever the current count reaches 0. If this bit is set, the reload is additionally qualified with the condition that the last word of the burst is being sent. This will delay the reload of the counter by only a few cycles from when the counter value reaches 0. 0 = DISABLE 1 = ENABLE
15:0	0x0	COUNT_VALUE: DMA COUNT Init/Reload Value.

19.5.3.5 AHBDMA_IRQ_STA_CPU_0

AHB-DMA MASK_CPU Register Usage

Gathers all the after-masking CPU directed IRQ status bits.

Offset: 014h | Read/Write: RO | Reset: 0bxxxx

Bit	Reset	Description
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

19.5.3.6 AHBDMA_IRQ_STA_COP_0

AHB-DMA MASK_COP Register Usage

Gathers all the after-masking COP directed IRQ status bits.

Offset: 018h | Read/Write: RO | Reset: 0bxxxx

Bit	Reset	Description
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

19.5.3.7 AHBDMA_IRQ_MASK_0

AHB-DMA SET_MASK Register Usage

Allows the IRQ to propagate when enabled, this is the ANDed result of set and clear mask bits.

Offset: 01ch | Read/Write: RO | Reset: 0bxxxx

Bit	Reset	Description
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

19.5.3.8 AHBDMA_IRQ_MASK_SET_0

AHB-DMA CLR_MASK Register Usage

This register sets the Mask for the IRQs.

Offset: 020h | Read/Write: WO | Reset: 0b0000

Bit	Reset	Description
3	0x0	CH3: Writing 1 Sets the Mask Register for CH3 0 = DISABLE 1 = ENABLE
2	0x0	CH2: Writing 1 Sets the Mask Register for CH2 0 = DISABLE 1 = ENABLE
1	0x0	CH1: Writing 1 Sets the Mask Register for CH1 0 = DISABLE 1 = ENABLE
0	0x0	CH0: Writing 1 Sets the Mask Register for CH0 0 = DISABLE 1 = ENABLE

19.5.3.9 AHBDMA_IRQ_MASK_CLR_0

AHB-DMA Requestor Usage

This register clears the IRQs.

Offset: 024h | Read/Write: WO | Reset: 0b0000

Bit	Reset	Description
3	0x0	CH3: Writing 1 Clears the Mask Register for CH3 0 = DISABLE 1 = ENABLE
2	0x0	CH2: Writing 1 Clears the Mask Register for CH2 0 = DISABLE 1 = ENABLE
1	0x0	CH1: Writing 1 Clears the Mask Register for CH1 0 = DISABLE 1 = ENABLE
0	0x0	CH0: Writing 1 Clears the Mask Register for CH0 0 = DISABLE 1 = ENABLE

19.5.3.10 AHBDMA_RDWR_COHERENCY_0

0x6000_8028: AHBDMA outstanding R/W per channel.

bit[31] is the coherency status on the AHB-side of the AHBDMA.

When MST_GIZMO_WRQUE_EMPTY is active, there are no writeAHB transactions from any AHBDMA channel on their way out.

bits[19:16] and [3:0] check the status of WriteXMB and ReadXMB transactions on the direct-to-memory CIF side of the AHBDMA.

Offset: 028h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	MST_GIZMO_WRQUE_EMPTY : When active, there are no write AHB transactions
19	X	CH3_RDS_ACTIVE: 1=ahbdma channel 3 has read transaction(s) outstanding
18	X	CH2_RDS_ACTIVE
17	X	CH1_RDS_ACTIVE
16	X	CH0_RDS_ACTIVE
3	X	CH3_WRS_ACTIVE: 1=ahbdma channel 3 has write transaction(s) outstanding
2	X	CH2_WRS_ACTIVE
1	X	CH1_WRS_ACTIVE
0	X	CH0_WRS_ACTIVE

19.5.3.11 AHBDMA_TEST_BUS_0

0x6000_802C: AHBDMA_TEST_BUS. This is a legacy observability bus.

Offset: 02ch | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	MSTGIZMO_WRQUE_CH3_PONG_STATUS
6	X	MSTGIZMO_WRQUE_CH3_PING_STATUS
5	X	MSTGIZMO_WRQUE_CH2_PONG_STATUS
4	X	MSTGIZMO_WRQUE_CH2_PING_STATUS
3	X	MSTGIZMO_WRQUE_CH1_PONG_STATUS
2	X	MSTGIZMO_WRQUE_CH1_PING_STATUS
1	X	MSTGIZMO_WRQUE_CH0_PONG_STATUS
0	X	MSTGIZMO_WRQUE_CH0_PING_STATUS

19.5.3.12 AHBDMA_PPCS_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 2nd-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: 040h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxx0000

Bit	Reset	Description
17	0x0	PPCS_RCLK_OVERRIDE
16	0x0	PPCS_WCLK_OVERRIDE
3	DISABLE	PPCS_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	PPCS_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	PPCS_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	PPCS_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

19.5.3.13 AHBDMA_TIMEOUT_WCOAL_PPCS_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: 044h | Read/Write: R/W | Reset: 0b0011001000110010

Bit	Reset	Description
15:8	0x32	PPCSAHBDMAW_WCOAL_TMVAL
7:0	0x32	PPCSAHBSLVW_WCOAL_TMVAL

19.5.4 AHBDMACHANNEL_REGISTERS

The description given for AHB-DMA-CHANNEL-0 applies to all the corresponding registers for the remaining channels AHB-DMA-CHANNEL-1 to AHB-DMA-CHANNEL-3

19.5.4.1 AHBDMACHAN_CHANNEL_0_CSR_0

AHB-DMA-CHANNEL-0 Control Register

Writing a 1 to bit [31] of an AHB DMA Channel Control Register will initiate the AHB 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 AHB DMA channel be programmed before writing into control register.

In "Once" mode, the channel is disabled after the AHB 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: 000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: 0 = Disable the DMA Channel 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: 0 = NOP or waiting 0 = DISABLE 1 = ENABLE
29	0x0	FL_BTWN: 0 = do not flush 0 = DISABLE 1 = ENABLE
28	0x0	HOLD: 0 = Disable 0 = DISABLE 1 = ENABLE
27	0x0	DIR: 0 = XMB read to AHB write 0 = DISABLE 1 = ENABLE
26	0x0	ONCE: 0 = Run for Multiple Block Transfer 0 = DISABLE 1 = ENABLE
25	0x0	TRIG: 0 = Independent of Trigger (no trigger set) 0 = DISABLE 1 = ENABLE
24	0x0	FLOW: 0 = Independent of DRQ request (no flow set) 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23:20	0x0	TRIG_SEL: 0 = SMP_30_same_as_bit_14 1 = SMP_31_same_as_bit_15 2 = HRQ2_XRQ_C 3 = HRQ3_XRQ_D 4 = HRQ4_N_A 5 = HRQ5_N_A 6 = HRQ6_TMR1 7 = HRQ7_TMR2 8 = AHB_0 9 = AHB_1 10 = AHB_2 11 = AHB_3 12 = SMP_28 13 = SMP_29 14 = SMP_30 15 = SMP_31
19:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = SMP_17 2 = SMP_18 3 = SMP_19 4 = SMP_20 5 = SMP_21 6 = SMP_22 7 = SMP_23 8 = SMP_24 9 = SMP_25 10 = SMP_26 11 = Host1x 12 = SRQ0_N_A 13 = SRQ1_N_A 14 = SRQ0_XRQ_A 15 = SRQ1_XRQ_B
15:2	0x0	WCOUNT: Number of 32 bit word cycles. This is encoded as N+1, so that if a value of 0 is programmed here, 1 32-bit word will be transferred.

19.5.4.2 AHBDMACHAN_CHANNEL_0_STA_0

AHB-DMA-CHANNEL-0 Status Register

Offset: 004h | Read/Write: R/W | Reset: 0b00x0xxxxxxxxxxx00000000000000

Bit	Reset	Description
31	0x0	BSY: 0 = NOP Read-only 0 = DISABLE 1 = ENABLE
30	0x0	IS_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
28	0x0	HALT: 0 = NOP (holding status) Read-only 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
15:2	0x0	COUNT: Remaining transfer count in 32bit word cycles Flags set / cleared by HW.

19.5.4.3 AHBDMACHAN_CHANNEL_0_AHB_PTR_0

AHB-DMA-CHANNEL-0 AHB Starting Address Pointer Register

If stride is enabled, the AHB address should be aligned to burst size.

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: AHB-DMA starting address for internal AHB Bus

19.5.4.4 AHBDMACHAN_CHANNEL_0_AHB_SEQ_0

AHB-DMA-CHANNEL-0 AHB Address Sequencer Register

An AHB DMA stride is a sub-section of an AHB DMA transfer, can be used in once mode, and is composed of one or more AHB DMA bursts. When stride is enabled, wrap should not be enabled. The size of a stride is programmable, but is always an integer multiple of 4 words.

The size of the AHB DMA stride must always be an integer multiple of the AHB DMA burst size. The size of the AHB DMA transfer count (total words in the transfer) must also be a multiple of both the burst size and the stride size.

If stride is enabled, the AHB_STRIDE should be aligned to burst size.

Offset: 014h | Read/Write: R/W | Reset: 0b0xxx0010000x000000000000000000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
27	0x0	AHB_DATA_SWAP: 0 = no data conversion, when 1 and writeAhb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x2	AHB_BURST: 0,1,5-7 = rsvd 2 = WORD 3 = FOUR_WORD 4 = EIGHT_WORD
23:21	0x0	AHB_STRIDE_SIZE: 0 = disabled, otherwise stride is enabled 0 = DISABLE 1 = WORDS_4 2 = WORDS_8 3 = WORDS_16 4 = WORDS_32 5 = WORDS_64 6 = WORDS_128 7 = WORDS_256
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default), reload each time 0 = DISABLE 1 = ENABLE
18	0x0	AHB_ADDR_WRAP: 0 = disable 0 = DISABLE 1 = ENABLE
17:2	0x0	AHB_STRIDE: This field tells us number of words that are supposed to be skipped on AHB side.

19.5.4.5 AHBDMACHAN_CHANNEL_0_XMB_PTR_0

AHB-DMA-CHANNEL-0 XMB Starting Address Pointer Register

The starting XMB address for the AHB DMA transfer must be aligned to an address boundary that is a multiple of the burst size.

Offset: 018h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	XMB_BASE: AHB-DMA Starting address for internal AHB Bus

19.5.4.6 AHBDMACHAN_CHANNEL_0_XMB_SEQ_0

AHB-DMA-CHANNEL-0 XMB Address Sequencer Register

If the stride is enabled, then the XMB_STRIDE must also be a multiple of the burst size.

Offset: 01ch | Read/Write: R/W | Reset: 0b0xxxxxxx000000000000000000

Bit	Reset	Description
27	0x0	XMB_DATA_SWAP: 0 = no data conversion, when 1 and writeXmb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default) (reload each time) 0 = DISABLE 1 = ENABLE
18	0x0	XMB_ADDR_WRAP: 0 disable 0 = DISABLE 1 = ENABLE
17:2	0x0	XMB_STRIDE: This field tells us number of words that are supposed to be skipped on XMB side.

19.5.4.7 AHBDMACHAN_CHANNEL_1_CSR_0

AHB-DMA-CHANNEL-1 Control Register

Offset: 020h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: 0 = Disable the DMA Channel 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: 0 = NOP 0 = DISABLE 1 = ENABLE
29	0x0	FL_BTWN: 0 = do not flush 0 = DISABLE 1 = ENABLE
28	0x0	HOLD: 0 = Disable 0 = DISABLE 1 = ENABLE
27	0x0	DIR: 0 = XMB read to AHB write 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
26	0x0	ONCE: 0 = Run for Multiple Block Transfer 0 = DISABLE 1 = ENABLE
25	0x0	TRIG: 0 = Independent of Trigger 0 = DISABLE 1 = ENABLE
24	0x0	FLOW: 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE
23:20	0x0	TRIG_SEL: 0 = SMP_30_same_as_bit_14 1 = SMP_31_same_as_bit_15 2 = HRQ2_XRQ_C 3 = HRQ3_XRQ_D 4 = HRQ4_N_A 5 = HRQ5_N_A 6 = HRQ6_TMR1 7 = HRQ7_TMR2 8 = AHB_0 9 = AHB_1 10 = AHB_2 11 = AHB_3 12 = SMP_28 13 = SMP_29 14 = SMP_30 15 = SMP_31
19:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = SMP_17 2 = SMP_18 3 = SMP_19 4 = SMP_20 5 = SMP_21 6 = SMP_22 7 = SMP_23 8 = SMP_24 9 = SMP_25 10 = SMP_26 11 = Host1x 12 = SRQ0_N_A 13 = SRQ1_N_A 14 = SRQ0_XRQ_A 15 = SRQ1_XRQ_B
15:2	0x0	WCOUNT: Number of 32 bit word cycles. This is encoded as N+1, so that if a value of 0 is programmed here, 1 32-bit word will be transferred.

19.5.4.8 AHBDMACHAN_CHANNEL_1_STA_0

AHB-DMA-CHANNEL-1 Status Register

Offset: 024h | Read/Write: R/W | Reset: 0b00x0xxxxxxxxxxx00000000000000

Bit	Reset	Description
31	0x0	BSY: 0 = NOP Read-only 0 = DISABLE 1 = ENABLE
30	0x0	IS_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR

Bit	Reset	Description
28	0x0	HALT: 0 = NOP 0 = DISABLE 1 = ENABLE
15:2	0x0	COUNT: Remaining transfer count in 32bit word cycles Flags set / cleared by HW

19.5.4.9 AHBDMACHAN_CHANNEL_1_AHB_PTR_0

AHB-DMA-CHANNEL-1 AHB Starting Address Pointer Register

Offset: 030h | Read/Write: R/W |Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: AHB-DMA starting address pointer for internal AHB Bus

19.5.4.10 AHBDMACHAN_CHANNEL_1_AHB_SEQ_0

AHB-DMA-CHANNEL-1 AHB Address Sequencer Register

Offset: 034h | Read/Write: R/W | Reset: 0b0xxx0010000x000000000000000000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
27	0x0	AHB_DATA_SWAP: 0 = no data conversion, when 1 and writeAhb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} enum (DISABLE=0x0,ENABLE=0x1)
26:24	0x2	AHB_BURST: 0,1,5-7 = rsvd 2 = WORD 3 = FOUR_WORD 4 = EIGHT_WORD
23:21	0x0	AHB_STRIDE_SIZE: 0 = disabled, otherwise stride is enabled 0 = DISABLE 1 = WORDS_4 2 = WORDS_8 3 = WORDS_16 4 = WORDS_32 5 = WORDS_64 6 = WORDS_128 7 = WORDS_256
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default), reload each time 0 = DISABLE 1 = ENABLE
18	0x0	AHB_ADDR_WRAP: 0 = disable 0 = DISABLE 1 = ENABLE
17:2	0x0	AHB_STRIDE: This field tells us number of words that are supposed to be skipped on AHB side.

19.5.4.11 AHBDMACHAN_CHANNEL_1_XMB_PTR_0

AHB-DMA-CHANNEL-1 XMB Starting Address Pointer Register

Offset: 038h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	XMB_BASE: AHB-DMA Starting address pointer for internal AHB Bus

19.5.4.12 AHBDMACHAN_CHANNEL_1_XMB_SEQ_0

AHB-DMA-CHANNEL-1 XMB Address Sequencer Register

Offset: 03ch | Read/Write: R/W | Reset: 0b0xxxxxxx000000000000000000

Bit	Reset	Description
27	0x0	XMB_DATA_SWAP: 0 = no data conversion, when 1 and writeXmb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default), reload each time 0 = DISABLE 1 = ENABLE
18	0x0	XMB_ADDR_WRAP: 1 enable: 0 disable ; When enabled the Address on XMB gets wrapped to same address. 0 = DISABLE 1 = ENABLE
17:2	0x0	XMB_STRIDE: This field tells us number of words that are supposed to be skipped on XMB side.

19.5.4.13 AHBDMACHAN_CHANNEL_2_CSR_0

AHB-DMA-CHANNEL-2 Control Register

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: 0 = Disable the DMA Channel 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: 0 = NOP or waiting 0 = DISABLE 1 = ENABLE
29	0x0	FL_BTWN: 0 = do not flush 0 = DISABLE 1 = ENABLE
28	0x0	HOLD: 0 = Disable 0 = DISABLE 1 = ENABLE
27	0x0	DIR: 0 = XMB read to AHB write 0 = DISABLE 1 = ENABLE
26	0x0	ONCE: 0 = Run for Multiple Block Transfer 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
25	0x0	TRIG: 0 = Independent of Trigger (no trigger set) 0 = DISABLE 1 = ENABLE
24	0x0	FLOW: 0 = Independent of DRQ request (no flow set) 0 = DISABLE 1 = ENABLE
23:20	0x0	TRIG_SEL: 0 = SMP_30_same_as_bit_14 1 = SMP_31_same_as_bit_15 2 = HRQ2_XRQ_C 3 = HRQ3_XRQ_D 4 = HRQ4_N_A 5 = HRQ5_N_A 6 = HRQ6_TMR1 7 = HRQ7_TMR2 8 = AHB_0 9 = AHB_1 10 = AHB_2 11 = AHB_3 12 = SMP_28 13 = SMP_29 14 = SMP_30 15 = SMP_31
19:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = SMP_17 2 = SMP_18 3 = SMP_19 4 = SMP_20 5 = SMP_21 6 = SMP_22 7 = SMP_23 8 = SMP_24 9 = SMP_25 10 = SMP_26 11 = Host1x 12 = SRQ0_N_A 13 = SRQ1_N_A 14 = SRQ0_XRQ_A 15 = SRQ1_XRQ_B
15:2	0x0	WCOUNT: Number of 32 bit word cycles. This is encoded as N+1, so that if a value of 0 is programmed here, 1 32-bit word will be transferred.

19.5.4.14 AHBDMACHAN_CHANNEL_2_STA_0

AHB-DMA-CHANNEL-2 Status Register

Offset: 044h | Read/Write: R/W | Reset: 0b00x0xxxxxxxxxxx00000000000000

Bit	Reset	Description
31	0x0	BSY: 0 = NOP Read-only 0 = DISABLE 1 = ENABLE
30	0x0	IS_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
28	0x0	HALT: 0 = NOP (holding status) Read-only 0 = DISABLE 1 = ENABLE
15:2	0x0	COUNT: Remaining transfer count in 32bit word cycles Flags set / cleared by HW.

19.5.4.15 AHBDMACHAN_CHANNEL_2_AHB_PTR_0

AHB-DMA-CHANNEL-2 AHB Starting Address Pointer Register

Offset: 050h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: AHB-DMA starting address for internal AHB Bus

19.5.4.16 AHBDMACHAN_CHANNEL_2_AHB_SEQ_0

AHB-DMA-CHANNEL-2 AHB Address Sequencer Register

Offset: 054h | Read/Write: R/W | Reset: 0b0xxx0010000x000000000000000000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
27	0x0	AHB_DATA_SWAP: 0 = no data conversion, when 1 and writeAhb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x2	AHB_BURST: 0,1,5-7 = rsvd 2 = WORD 3 = FOUR_WORD 4 = EIGHT_WORD
23:21	0x0	AHB_STRIDE_SIZE: 0 = disabled, otherwise stride is enabled 0 = DISABLE 1 = WORDS_4 2 = WORDS_8 3 = WORDS_16 4 = WORDS_32 5 = WORDS_64 6 = WORDS_128 7 = WORDS_256
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default), reload each time 0 = DISABLE 1 = ENABLE
18	0x0	AHB_ADDR_WRAP: 0 = disable 0 = DISABLE 1 = ENABLE
17:2	0x0	AHB_STRIDE: This field tells us number of words that are supposed to be skipped on AHB side.

19.5.4.17 AHBDMACHAN_CHANNEL_2_XMB_PTR_0

AHB-DMA-CHANNEL-2 XMB Starting Address Pointer Register

Offset: 058h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	XMB_BASE: AHB-DMA Starting address for internal AHB Bus

19.5.4.18 AHBDMACHAN_CHANNEL_2_XMB_SEQ_0

AHB-DMA-CHANNEL-2 XMB Address Sequencer Register

Offset: 05ch | Read/Write: R/W | Secure: Unprotected | Reset: 0b0xxxxxxx00000000000000000000 | Default: 0000.0000

Bit	Reset	Description
27	0x0	XMB_DATA_SWAP: 0 = no data conversion, when 1 and writeXmb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (def) eload each time) 0 = DISABLE 1 = ENABLE
18	0x0	XMB_ADDR_WRAP: 1 enable: 0 disable ; When enabled the Address on XMB gets wrapped to same address. 0 = DISABLE 1 = ENABLE
17:2	0x0	XMB_STRIDE: This field tells us number of words that are supposed to be skipped on XMB side.

19.5.4.19 AHBDMACHAN_CHANNEL_3_CSR_0

AHB-DMA-CHANNEL-3 Control Register

Offset: 060h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: 0 = Disable the DMA Channel 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: 0 = NOP 0 = DISABLE 1 = ENABLE
29	0x0	FL_BTWN: 0 = do not flush 0 = DISABLE 1 = ENABLE
28	0x0	HOLD: 0 = Disable 0 = DISABLE 1 = ENABLE
27	0x0	DIR: 0 = XMB read to AHB write 0 = DISABLE 1 = ENABLE
26	0x0	ONCE: 0 = Run for Multiple Block Transfer 0 = DISABLE 1 = ENABLE
25	0x0	TRIG: 0 = Independent of Trigger 0 = DISABLE 1 = ENABLE
24	0x0	FLOW: 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
23:20	0x0	TRIG_SEL: 0 = SMP_30_same_as_bit_14 1 = SMP_31_same_as_bit_15 2 = HRQ2_XRQ_C 3 = HRQ3_XRQ_D 4 = HRQ4_N_A 5 = HRQ5_N_A 6 = HRQ6_TMR1 7 = HRQ7_TMR2 8 = AHB_0 9 = AHB_1 10 = AHB_2 11 = AHB_3 12 = SMP_28 13 = SMP_29 14 = SMP_30 15 = SMP_31
19:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = SMP_17 2 = SMP_18 3 = SMP_19 4 = SMP_20 5 = SMP_21 6 = SMP_22 7 = SMP_23 8 = SMP_24 9 = SMP_25 10 = SMP_26 11 = Host1x 12 = SRQ0_N_A 13 = SRQ1_N_A 14 = SRQ0_XRQ_A 15 = SRQ1_XRQ_B
19:16	0x0	REQ_SEL: Lower 16 Requestors (15-0)
15:2	0x0	WCOUNT: Number of 32 bit word cycles. This is encoded as N+1, so that if a value of 0 is programmed here, 1 32-bit word will be transferred.

19.5.4.20 AHBDMACHAN_CHANNEL_3_STA_0

AHB-DMA-CHANNEL-3 Status Register

Offset: 064h | Read/Write: R/W | Reset: 0b00x0xxxxxxxxxxx00000000000000

Bit	Reset	Description
31	0x0	BSY: 0 = NOP Read-only 0 = DISABLE 1 = ENABLE
30	0x0	IS_EOC: Write '1' to clear the flag 0 = NO_INTR 1 = INTR
28	0x0	HALT: 0 = NOP 0 = DISABLE 1 = ENABLE
15:2	0x0	COUNT: Remaining transfer count in 32bit word cycles Flags set / cleared by HW

19.5.4.21 AHBDMACHAN_CHANNEL_3_AHB_PTR_0

AHB-DMA-CHANNEL-3 AHB Starting Address Pointer Register

Offset: 070h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: AHB-DMA starting address pointer for internal AHB Bus

19.5.4.22 AHBDMACHAN_CHANNEL_3_AHB_SEQ_0

AHB-DMA-CHANNEL-3 AHB Address Sequencer Register

Offset: 074h | Read/Write: R/W | Reset: 0b0xxx0010000x000000000000000000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 1 = CPU 0 = COP
27	0x0	AHB_DATA_SWAP: 0 = no data conversion, when 1 and writeAhb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
26:24	0x2	AHB_BURST: 0,1,5-7 = rsvd 2 = WORD 3 = FOUR_WORD 4 = EIGHT_WORD
23:21	0x0	AHB_STRIDE_SIZE: 0 = disabled, otherwise stride is enabled 0 = DISABLE 1 = WORDS_4 2 = WORDS_8 3 = WORDS_16 4 = WORDS_32 5 = WORDS_64 6 = WORDS_128 7 = WORDS_256
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default), reload each time 0 = DISABLE 1 = ENABLE
18	0x0	AHB_ADDR_WRAP: 0 = disable 0 = DISABLE 1 = ENABLE
17:2	0x0	AHB_STRIDE: This field tells us number of words that are supposed to be skipped on AHB side.

19.5.4.23 AHBDMACHAN_CHANNEL_3_XMB_PTR_0

AHB-DMA-CHANNEL-3 XMB Starting Address Pointer Register

Offset: 078h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	XMB_BASE: AHB-DMA Starting address pointer for internal AHB Bus

19.5.4.24 AHBDMACHAN_CHANNEL_3_XMB_SEQ_0

AHB-DMA-CHANNEL-3 XMB Address Sequencer Register

Offset: 07ch | Read/Write: R/W | Reset: 0b0xxxxxxx000000000000000000

Bit	Reset	Description
27	0x0	XMB_DATA_SWAP: 0 = no data conversion, when 1 and writeXmb: [31:0]->{[7:0],[15:8],[23:16],[31:24]} 0 = DISABLE 1 = ENABLE
19	0x0	DBL_BUF: 0 = Reload Base Address for 1X blocks (default), reload each time 0 = DISABLE 1 = ENABLE
18	0x0	XMB_ADDR_WRAP: 0 = disable 0 = DISABLE 1 = ENABLE
17:2	0x0	XMB_STRIDE: This field tells us number of words that are supposed to be skipped on XMB side.

[THIS PAGE INTENTIONALLY LEFT BLANK]

20.0 APB

20.1 APB Miscellaneous Registers

20.1.1 Strap Registers

This register allows the state of the HW straps on the board to be read. These strap values are captured at reset.

20.1.1.1 APB_MISC_PP_STRAPPING_OPT_A_0

Strapping Options Register

Offset: 008h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxx0xxxxxxx0

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_clk,gmi_adv_n} 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
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 In emulation (HIDREV_MAJORREV==0), this field indicates the RAM type connected. For QT (HIDREV_MINORREV==0): 0=SIM, 1=DDR, 2=DDR2, 3=LPDDR2 For FPGA (HIDREV_MINORREV==1): 0=SIM, 1=DDR, 2=DDR2, 3=LPDDR2
0	RSVD1	NOR_WIDTH: 0 = RSVD1 1 = RSVD2

20.1.2 JTAG Configuration Register

20.1.2.1 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: 024h | Read/Write: R/W | Reset: 0b01xxxx00

Bit	Reset	Description
7	DISABLE	TBE: 0 = Disable ; 1 = Enable RTCK Daisychaining 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

20.1.3 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 internal pull-up when pad has external pull down and vice versa will cause significant increase in power consumption.

Table 31. 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

20.1.3.1 APB_MISC_PP_PULLUPDOWN_REG_C_0

Offset: 0a8h | Read/Write: R/W | Reset: 0b000000

Bit	Reset	Description
31:30	NORMAL	XM2C_PU_PD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD

Bit	Reset	Description
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

20.1.3.2 APB_MISC_SC1X_PADS_VIP_VCLKCTRL_0

Enable VCLK pad to get external clock for VI.

VCLK Control Register

Offset: 428h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	0x0	INVERSION: VCLK invert enable 0 = DISABLE 1 = ENABLE
0	0x0	IE: VCLK input enable 0 = DISABLE 1 = ENABLE

20.1.3.3 APB_MISC_SC1X_PADS_TVDAC_VHSYNCCTRL_0

In VGA mode of operation, there are separate HSYNC and VSYNC sync outputs to go with the R/G/B analog output signals.

TVDAC_HSYNCDLY and TVDAC_VSYNCDLY specify additional signal delay to be applied to HSYNC and VSYNC outputs. Values {0, 1, 2, 3} correspond to the added delay in {0-, 1-, 2-, 3-} pixel periods.

TVDAC HSYNC & VSYNC delay control

Offset: 438h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3:2	0x0	TVDAC_VSYNCDLY
1:0	0x0	TVDAC_HSYNCDLY

20.1.3.4 APB_MISC_SC1X_PADS_TVDAC_CONTROL_0

- DAC_IDDQ, DAC_POWERDOWN, and DAC_DETECT_EN are power down control to all 3 DACs.
- DAC_SLEEPR, DAC_SLEEPG, and DAC_SLEEPB are individual power down controls.
- DAC_COMP_TH[1:0] is to choose threshold voltage of the internal comparator for load detection.
  {0, 1, 2, 3} for {0.24 V, 0.30 V, 0.48 V, 0.60 V}
- DAC_BGAP_AVG_ON is to enable bandgap averaging (default is 0).
- DAC_BGAP_CURVE[2:0] is to adjust temperature coefficient (default is 0x4).
- DAC_BGAP_CNTL[3:0] is to set up clock frequency in ATE calibration mode.

- DAC_ATEST[2:0] is for testing. To select various internal analog outputs to be sent to the VREF pin.
- DAC_CNTL[4:0] is to adjust bias current and bias resistance.
- DAC_COMPR_EN, DAC_COMPG_EN, and DAC_COMPB_EN are to enable comparator outputs.
- DAC_PLUG_OK is to indicate the load status. 0/1 for unloaded/loaded.

TV DAC control

Offset: 43ch | Read/Write: R/W | Reset: 0b0000x00000000000100000111011

Bit	Reset	Description
28	0x0	DAC_PLUG_OK: Indicate load status 0 = UNLOADED 1 = LOADED
27	0x0	DAC_COMPB_EN: Enable COMPOUTB output 0 = DISABLE 1 = ENABLE
26	0x0	DAC_COMPG_EN: Enable COMPOUTG output 0 = DISABLE 1 = ENABLE
25	0x0	DAC_COMPR_EN: Enable COMPOUTR output 0 = DISABLE 1 = ENABLE
23:19	0x0	DAC_CNTL: Reserved for additional control
18:16	0x0	DAC_ATEST: For debugging. Selects internal analog output to be sent out of VREF pin
15:12	0x0	DAC_BGAP_CNTL: Control bits for bandgap
11:9	0x4	DAC_BGAP_CURVE: To adjust temp coeff
8	0x0	DAC_BGAP_AVG_ON: Turn bandgap averaging on/off 0 = DISABLE 1 = ENABLE
7:6	0x0	DAC_COMP_TH: Adjust threshold voltage of comparator inside DAC
5	0x1	DAC_SLEEPB: Low power (sleep) mode. Shut down OUTB output 0 = DISABLE 1 = ENABLE
4	0x1	DAC_SLEEPG: Low power (sleep) mode. Shut down OUTG output 0 = DISABLE 1 = ENABLE
3	0x1	DAC_SLEEPR: Low power (sleep) mode. Shut down OUTR output 0 = DISABLE 1 = ENABLE
2	0x0	DAC_DETECT_EN: Power down everything including the band-gap 0 = DISABLE 1 = ENABLE
1	0x1	DAC_POWERDOWN: Power down everything except the band-gap 0 = DISABLE 1 = ENABLE
0	0x1	DAC_IDDQ: Power down everything including the band-gap 0 = DISABLE 1 = ENABLE

20.1.3.5 APB_MISC_SC1X_PADS_TVDAC_STATUS_0

Output signals from internal comparators

TV DAC status

Offset: 440h | Read/Write: RO | Reset: 0bxxx

Bit	Reset	Description
2	X	DAC_COMPOUTB: Channel B comparator output for auto-detect 0 = COMPINB > threshold 1 = COMPINB < threshold, or when POWERDOWN==1 comparison threshold = 0.325V
1	X	DAC_COMPOUTG: Channel G comparator output for auto-detect 0 = COMPING > threshold 1 = COMPING < threshold, or when POWERDOWN==1 comparison threshold = 0.325V
0	X	DAC_COMPOUTR: Channel R comparator output for auto-detect 0 = COMPINR > threshold 1 = COMPINR < threshold, or when POWERDOWN==1 comparison threshold = 0.325V

20.1.3.6 APB_MISC_SC1X_PADS_TVDAC_DINCONFIG_0

- DAC_FIFO_TH[2:0] is to specify number of pixels kept in the input FIFO.
- DAC_SOURCE[1:0] is to specify input signal sources to the TVDAC.

  The sources are TVO, DISPLAY, and DISPLAYB.
- DAC_DIN_ORIDE_EN is to enable DAC_DIN_ORIDE to override the input signal.
- DAC_DIN_ORIDE[9:0] is to override regular input signals for testing.
- DAC_AMPIN[7:0] is to set DAC output amplitude.

There are 3 ways that DAC's output amplitude range is calibrated. They are,

- SDTV mode (VREF=1.4V)
- CRT (VGA) mode (VREF=0.7V)
- HDTV mode (VREF=1.0V)

Corresponding calibration values for DAC_AMPIN are provided from FUSE module. They are,

- DAC_SDTV_CALIB[7:0]
- DAC_CRT_CALIB[7:0]
- DAC_HDTV_CALIB[7:0]

Corresponding to the DAC_SOURCE selection, those calibration values shall be selectively loaded to DAC_AMPIN.

DAC_SOURCE	DAC_AMPIN
0 (TVDAC_OFF)	(dont_care)
1 (TVO)	DAC_SDTV_CALIB[7:0]
2 (DISPLAY)	DAC_CRT_CALIB[7:0]
3 (DISPLAYB)	DAC_CRT_CALIB[7:0]

TV DAC FIFO config

Offset: 444h | Read/Write: R/W | Reset: 0b00000000000000000x0xx00x010

Bit	Reset	Description
27:20	0x0	DAC_AMPIN: AMPIN

Bit	Reset	Description
19:10	0x0	DAC_DIN_ORIDE: DIN override
8	0x0	DAC_DIN_ORIDE_EN: Override DAC DIN inputs 0 = DISABLE 1 = ENABLE
5:4	0x0	DAC_SOURCE: INPUT source for TVDAC 0 = TVDAC_OFF 1 = TVO 2 = DISPLAY 3 = DISPLAYB
2:0	0x2	DAC_FIFO_TH: Data Input FIFO threshold

20.1.3.7 APB_MISC_GP_HIDREV_0

Chip ID revision register

Offset: 804h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
19:16	X	MINORREV: Chip ID minor revision (IF MAJORREV==0(Emulation) THEN 0: QT, 1:E388 FPGA)
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

20.1.4 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 32. Low Power Mode Functions

LPMD1	LPMD0	Power Mode Selected
0	0	X/8 Current Setting. Lowest Power or Current (8*Z ohms)
0	1	X/4 Current Setting (4*Z ohms)
1	0	X/2 Current Setting (2*Z ohms)
1	1	X Current Setting (Z ohm = 50 ohms). 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.

20.1.4.1 APB_MISC_GP_AOCFG1PADCTRL_0

AOCFG1 Pad Control Register

Offset: 868h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_AOCFG1_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_AOCFG1_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_AOCFG1_CAL_DRVUP
16:12	0x12	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 schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_AOCFG1_HSM_EN: AOCFG1 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.2 APB_MISC_GP_AOCFG2PADCTRL_0

AOCFG2 Pad Control Register

Offset: 86ch | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_AOCFG2_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_AOCFG2_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_AOCFG2_CAL_DRVUP
16:12	0x12	CFG2TMC_AOCFG2_CAL_DRVDN
5:4	0x3	CFG2TMC_AOCFG2_LPMD: AOCFG2 data pins low power mode select
3	0x0	CFG2TMC_AOCFG2_SCHMT_EN: AOCFG2 data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_AOCFG2_HSM_EN: AOCFG2 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.3 APB_MISC_GP_ATCFG1PADCTRL_0

ATCFG1 Pad Control Register

Offset: 870h | Read/Write: R/W | Reset: 0b001100111011010010xxxxxxx1100

Bit	Reset	Description
31:28	0x3	CFG2TMC_ATCFG1_CAL_DRVUP_SLWF: Only bit[29:28] are valid for this config. bit[31:30] are redundant.
27:24	0x3	CFG2TMC_ATCFG1_CAL_DRVDN_SLWR: Only bit[25:24] are valid for this config. bit[27:26] are redundant
23:19	0x16	CFG2TMC_ATCFG1_CAL_DRVUP
18:14	0x12	CFG2TMC_ATCFG1_CAL_DRVDN
5:4	0x3	CFG2TMC_ATCFG1_LPMD: ATCFG1 data pins low power mode select
3	0x0	CFG2TMC_ATCFG1_SCHMT_EN: ATCFG1 data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG1_HSM_EN: ATCFG1 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.4 APB_MISC_GP_ATCFG2PADCTRL_0

ATCFG2 Pad Control Register

Offset: 874h | Read/Write: R/W | Reset: 0b001100111011010010xxxxxxxx1100

Bit	Reset	Description
31:28	0x3	CFG2TMC_ATCFG2_CAL_DRVUP_SLWF: Only bit[29:28] are valid for this config. bit[31:30] are redundant.
27:24	0x3	CFG2TMC_ATCFG2_CAL_DRVDN_SLWR: Only bit[25:24] are valid for this config. bit[27:26] are redundant.
23:19	0x16	CFG2TMC_ATCFG2_CAL_DRVUP
18:14	0x12	CFG2TMC_ATCFG2_CAL_DRVDN
5:4	0x3	CFG2TMC_ATCFG2_LPMD: ATCFG2 data pins low power mode select
3	0x0	CFG2TMC_ATCFG2_SCHMT_EN: ATCFG2 data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG2_HSM_EN: ATCFG2 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.5 APB_MISC_GP_ATCFG3PADCTRL_0

ATCFG3 Pad Control Register

Offset: 878h | Read/Write: R/W | Reset: 0b1111xxxx1011010010xxxxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_ATCFG3_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_ATCFG3_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_ATCFG3_CAL_DRVUP
18:14	0x12	CFG2TMC_ATCFG3_CAL_DRVDN
5:4	0x3	CFG2TMC_ATCFG3_LPMD: ATCFG3 data pins low power mode select
3	0x0	CFG2TMC_ATCFG3_SCHMT_EN: ATCFG3 data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG3_HSM_EN: ATCFG3 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.6 APB_MISC_GP_ATCFG4PADCTRL_0

ATCFG4 Pad Control Register

Offset: 87ch | Read/Write: R/W | Reset: 0b1111xxxx1011010010xxxxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_ATCFG4_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_ATCFG4_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_ATCFG4_CAL_DRVUP
18:14	0x12	CFG2TMC_ATCFG4_CAL_DRVDN
5:4	0x3	CFG2TMC_ATCFG4_LPMD: ATCFG4 data pins low power mode select
3	0x0	CFG2TMC_ATCFG4_SCHMT_EN: ATCFG4 data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG4_HSM_EN: ATCFG4 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.7 APB_MISC_GP_ATCFG5PADCTRL_0

ATCFG5 Pad Control Register

Offset: 880h | Read/Write: R/W | Reset: 0b1111xxxx1011010010xxxxxxxx1100

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 schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_ATCFG5_HSM_EN: ATCFG5 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.8 APB_MISC_GP_CDEV1CFGPADCTRL_0

CDEV1CFG Pad Control Register

Offset: 884h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_CDEV1CFG_CAL_DRVUP_SLWF

Bit	Reset	Description
29:28	0x3	CFG2TMC_CDEV1CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_CDEV1CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_CDEV1CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_CDEV1CFG_LPMD: CDEV1CFG data pins low power mode select
3	0x0	CFG2TMC_CDEV1CFG_SCHMT_EN: CDEV1CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CDEV1CFG_HSM_EN: CDEV1CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.9 APB_MISC_GP_CDEV2CFGPADCTRL_0

CDEV2CFG Pad Control Register

Offset: 888h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_CDEV2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_CDEV2CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_CDEV2CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_CDEV2CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_CDEV2CFG_LPMD: CDEV2CFG data pins low power mode select
3	0x0	CFG2TMC_CDEV2CFG_SCHMT_EN: CDEV2CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CDEV2CFG_HSM_EN: CDEV2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.10 APB_MISC_GP_CSUSCFGPADCTRL_0

CSUSCFG Pad Control Register

Offset: 88ch | Read/Write: R/W | Reset: 0b0011001110110xx10010

Bit	Reset	Description
31:28	0x3	CFG2TMC_CSUSCFG_CAL_DRVUP_SLWF
27:24	0x3	CFG2TMC_CSUSCFG_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_CSUSCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_CSUSCFG_CAL_DRVDN

20.1.4.11 APB_MISC_GP_DAP1CFGPADCTRL_0

DAP1CFG Pad Control Register

Offset: 890h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP1CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP1CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DAP1CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_DAP1CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DAP1CFG_LPMD: DAP1CFG data pins low power mode select
3	0x0	CFG2TMC_DAP1CFG_SCHMT_EN: DAP1CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP1CFG_HSM_EN: DAP1CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.12 APB_MISC_GP_DAP2CFGPADCTRL_0

DAP2CFG Pad Control Register

Offset: 894h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP2CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DAP2CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_DAP2CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DAP2CFG_LPMD: DAP2CFG data pins low power mode select
3	0x0	CFG2TMC_DAP2CFG_SCHMT_EN: DAP2CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP2CFG_HSM_EN: DAP2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.13 APB_MISC_GP_DAP3CFGPADCTRL_0

DAP3CFG Pad Control Register

Offset: 898h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP3CFG_CAL_DRVUP_SLWF

Bit	Reset	Description
29:28	0x3	CFG2TMC_DAP3CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DAP3CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_DAP3CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DAP3CFG_LPMD: DAP3CFG data pins low power mode select
3	0x0	CFG2TMC_DAP3CFG_SCHMT_EN: DAP3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP3CFG_HSM_EN: DAP3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.14 APB_MISC_GP_DAP4CFGPADCTRL_0

DAP4CFG Pad Control Register

Offset: 89ch | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DAP4CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DAP4CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DAP4CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_DAP4CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DAP4CFG_LPMD: DAP4CFG data pins low power mode select
3	0x0	CFG2TMC_DAP4CFG_SCHMT_EN: DAP4CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DAP4CFG_HSM_EN: DAP4CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.15 APB_MISC_GP_DBGCFGPADCTRL_0

DBGCFG Pad Control Register

Offset: 8a0h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DBGCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DBGCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DBGCFG_CAL_DRVUP
16:12	0x12	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 schmidt enable

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DBGCFG_HSM_EN: DBGCFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.16 APB_MISC_GP_LCDCFG1PADCTRL_0

LCDCFG1 Pad Control Register

Offset: 8a4h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_LCDCFG1_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_LCDCFG1_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_LCDCFG1_CAL_DRVUP
16:12	0x12	CFG2TMC_LCDCFG1_CAL_DRVDN
5:4	0x3	CFG2TMC_LCDCFG1_LPMD: LCDCFG1 data pins low power mode select
3	0x0	CFG2TMC_LCDCFG1_SCHMT_EN: LCDCFG1 data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_LCDCFG1_HSM_EN: LCDCFG1 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.17 APB_MISC_GP_LCDCFG2PADCTRL_0

LCDCFG2 Pad Control Register

Offset: 8a8h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_LCDCFG2_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_LCDCFG2_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_LCDCFG2_CAL_DRVUP
16:12	0x12	CFG2TMC_LCDCFG2_CAL_DRVDN
5:4	0x3	CFG2TMC_LCDCFG2_LPMD: LCDCFG2 data pins low power mode select
3	0x0	CFG2TMC_LCDCFG2_SCHMT_EN: LCDCFG2 data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_LCDCFG2_HSM_EN: LCDCFG2 data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.18 APB_MISC_GP_SDIO2CFGPADCTRL_0

SDIO2CFG Pad Control Register

Offset: 8ach | Read/Write: R/W | Reset: 0b1111x0010110x0010010xxxxxxxx00

Bit	Reset	Description
31:30	0x3	CFG2TMC_SDIO2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_SDIO2CFG_CAL_DRVDN_SLWR
26:20	0x16	CFG2TMC_SDIO2CFG_CAL_DRVUP
18:12	0x12	CFG2TMC_SDIO2CFG_CAL_DRVDN

3	0x0	CFG2TMC_SDIO2CFG_SCHMT_EN: SDIO2CFG data pins schmidt enable 5:4 rw
		CFG2TMC_SDIO2CFG_LPMD i=0x3 // SDIO2CFG data pins low power mode select
		0 = DISABLE
		1 = ENABLE

2	0x0	CFG2TMC_SDIO2CFG_HSM_EN: SDIO2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.19 APB_MISC_GP_SDIO3CFGPADCTRL_0

SDIO3CFG Pad Control Register

Offset: 8b0h | Read/Write: R/W | Reset: 0b1111x0010110x0010010xxxxxxxx00

Bit	Reset	Description
31:30	0x3	CFG2TMC_SDIO3CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_SDIO3CFG_CAL_DRVDN_SLWR
26:20	0x16	CFG2TMC_SDIO3CFG_CAL_DRVUP
18:12	0x12	CFG2TMC_SDIO3CFG_CAL_DRVDN

3	0x0	CFG2TMC_SDIO3CFG_SCHMT_EN: SDIO3CFG data pins schmidt enable 5:4 rw
		CFG2TMC_SDIO3CFG_LPMD i=0x3 // SDIO3CFG data pins low power mode select
		0 = DISABLE
		1 = ENABLE

2	0x0	CFG2TMC_SDIO3CFG_HSM_EN: SDIO3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.20 APB_MISC_GP_SPICFGPADCTRL_0

SPICFG Pad Control Register

Offset: 8b4h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_SPICFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_SPICFG_CAL_DRVDN_SLWR

Bit	Reset	Description
24:20	0x16	CFG2TMC_SPICFG_CAL_DRVUP
16:12	0x12	CFG2TMC_SPICFG_CAL_DRVDN
5:4	0x3	CFG2TMC_SPICFG_LPMD: SPICFG data pins low power mode select
3	0x0	CFG2TMC_SPICFG_SCHMT_EN: SPICFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_SPICFG_HSM_EN: SPICFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.21 APB_MISC_GP_UAACFGPADCTRL_0

UAACFG Pad Control Register

Offset: 8b8h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_UAACFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UAACFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_UAACFG_CAL_DRVUP
16:12	0x12	CFG2TMC_UAACFG_CAL_DRVDN
5:4	0x3	CFG2TMC_UAACFG_LPMD: UAACFG data pins low power mode select
3	0x0	CFG2TMC_UAACFG_SCHMT_EN: UAACFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UAACFG_HSM_EN: UAACFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.22 APB_MISC_GP_UABCFGPADCTRL_0

UABCFG Pad Control Register

Offset: 8bch | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_UABCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UABCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_UABCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_UABCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_UABCFG_LPMD: UABCFG data pins low power mode select
3	0x0	CFG2TMC_UABCFG_SCHMT_EN: UABCFG data pins schmidt enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	0x0	CFG2TMC_UABCFG_HSM_EN: UABCFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.23 APB_MISC_GP_UART2CFGPADCTRL_0

UART2CFG Pad Control Register

Offset: 8c0h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_UART2CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UART2CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_UART2CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_UART2CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_UART2CFG_LPMD: UART2CFG data pins low power mode select
3	0x0	CFG2TMC_UART2CFG_SCHMT_EN: UART2CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UART2CFG_HSM_EN: UART2CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.24 APB_MISC_GP_UART3CFGPADCTRL_0

UART3CFG Pad Control Register

Offset: 8c4h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_UART3CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UART3CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_UART3CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_UART3CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_UART3CFG_LPMD: UART3CFG data pins low power mode select
3	0x0	CFG2TMC_UART3CFG_SCHMT_EN: UART3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UART3CFG_HSM_EN: UART3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.25 APB_MISC_GP_VICFG1PADCTRL_0

VICFG1 Pad Control Register

Offset: 8c8h | Read/Write: R/W | Reset: 0b001100111011010010

Bit	Reset	Description
31:28	0x3	CFG2TMC_VICFG1_CAL_DRVUP_SLWF
27:24	0x3	CFG2TMC_VICFG1_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_VICFG1_CAL_DRVUP
18:14	0x12	CFG2TMC_VICFG1_CAL_DRVDN

20.1.4.26 APB_MISC_GP_VIVTTGENPADCTRL_0

Offset: 8cch | Read/Write: R/W | Reset: 0b000xxxxx000x001x001xx1xxx00

Bit	Reset	Description
26:24	0x0	CFG2TMC_VICFG1_VTTGEN_DRVUP
18:16	0x0	CFG2TMC_VICFG1_VTTGEN_DRVDN
14:12	0x1	CFG2TMC_VICFG1_VTTGEN_VAUXP_LEVEL
10:8	0x1	CFG2TMC_VICFG1_VTTGEN_VCLAMP_LEVEL
5	0x1	CFG2TMC_VICFG1_VTTGEN_E_PWRD: Only need to enable if > 1.25V IO
1	0x0	CFG2TMC_VICFG1_VTTGEN_SHORT_PWRGND
0	0x0	CFG2TMC_VICFG1_VTTGEN_SHORT

20.1.4.27 APB_MISC_GP_SDIO1CFGPADCTRL_0

XM2 MISC/VTTGEN Pad Control Register

SDIO1CFG Pad control register

Offset: 8ech | Read/Write: R/W | Reset: 0b1111x0010110x0010010xxxxxxxx00

Bit	Reset	Description
31:30	0x3	CFG2TMC_SDIO1CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_SDIO1CFG_CAL_DRVDN_SLWR
26:20	0x16	CFG2TMC_SDIO1CFG_CAL_DRVUP
18:12	0x12	CFG2TMC_SDIO1CFG_CAL_DRVDN

3	0x0	CFG2TMC_SDIO1CFG_SCHMT_EN: data pins schmidt enable 5:4 rw
		CFG2TMC_SDIO1CFG_LPMD i=0x3 // data pins low power mode select
		0 = DISABLE
		1 = ENABLE

2	0x0	CFG2TMC_SDIO1CFG_HSM_EN: data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.28 APB_MISC_GP_CRTCFGPADCTRL_0

XM2CFGD Pad Control Register

CRTCFG Pad control register

Offset: 8f8h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_CRTCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_CRTCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_CRTCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_CRTCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_CRTCFG_LPMD: data pins low power mode select
3	0x0	CFG2TMC_CRTCFG_SCHMT_EN: data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CRTCFG_HSM_EN: data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.29 APB_MISC_GP_DDCCFGPADCTRL_0

DDCCFG Pad Control Register

Offset: 8fch | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DDCCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DDCCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DDCCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_DDCCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DDCCFG_LPMD: data pins low power mode select
3	0x0	CFG2TMC_DDCCFG_SCHMT_EN: data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DDCCFG_HSM_EN: data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.30 APB_MISC_GP_GMACFGPADCTRL_0

GMACFG Pad Control Register

Offset: 900h | Read/Write: R/W | Reset: 0b001100111011010010

Bit	Reset	Description
31:28	0x3	CFG2TMC_GMACFG_CAL_DRVUP_SLWF
27:24	0x3	CFG2TMC_GMACFG_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_GMACFG_CAL_DRVUP
18:14	0x12	CFG2TMC_GMACFG_CAL_DRVDN

20.1.4.31 APB_MISC_GP_GMBCFGPADCTRL_0

GMBCFG Pad Control Register

Offset: 904h | Read/Write: R/W | Reset: 0b001100111011010010

Bit	Reset	Description
31:28	0x3	CFG2TMC_GMBCFG_CAL_DRVUP_SLWF
27:24	0x3	CFG2TMC_GMBCFG_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_GMBCFG_CAL_DRVUP
18:14	0x12	CFG2TMC_GMBCFG_CAL_DRVDN

20.1.4.32 APB_MISC_GP_GMCCFGPADCTRL_0

GMCCFG Pad Control Register

Offset: 908h | Read/Write: R/W | Reset: 0b001100111011010010

Bit	Reset	Description
31:28	0x3	CFG2TMC_GMCCFG_CAL_DRVUP_SLWF
27:24	0x3	CFG2TMC_GMCCFG_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_GMCCFG_CAL_DRVUP
18:14	0x12	CFG2TMC_GMCCFG_CAL_DRVDN

20.1.4.33 APB_MISC_GP_GMDCFGPADCTRL_0

GMDCFG Pad Control Register

Offset: 90ch | Read/Write: R/W | Reset: 0b001100111011010010

Bit	Reset	Description
31:28	0x3	CFG2TMC_GMDCFG_CAL_DRVUP_SLWF
27:24	0x3	CFG2TMC_GMDCFG_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_GMDCFG_CAL_DRVUP

Bit	Reset	Description
18:14	0x12	CFG2TMC_GMDCFG_CAL_DRVDN

20.1.4.34 APB_MISC_GP_GMECFGPADCTRL_0

GMECFG Pad Control Register

Offset: 910h | Read/Write: R/W | Reset: 0b1111xxxx1011010010xxxxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMECFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMECFG_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_GMECFG_CAL_DRVUP
18:14	0x12	CFG2TMC_GMECFG_CAL_DRVDN
5:4	0x3	CFG2TMC_GMECFG_LPMD: data pins low power mode select
3	0x0	CFG2TMC_GMECFG_SCHMT_EN: data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMECFG_HSM_EN: data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.35 APB_MISC_GP_GMFCFGPADCTRL_0

GMFCFG Pad Control Register

Offset: 914h | Read/Write: R/W | Reset: 0b1111xxxx1011010010xxxxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMFCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMFCFG_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_GMFCFG_CAL_DRVUP
18:14	0x12	CFG2TMC_GMFCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_GMFCFG_LPMD: data pins low power mode select
3	0x0	CFG2TMC_GMFCFG_SCHMT_EN: data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMFCFG_HSM_EN: data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.36 APB_MISC_GP_GMGCFGPADCTRL_0

GMGCFG Pad Control Register

Offset: 918h | Read/Write: R/W | Reset: 0b1111xxxx1011010010xxxxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMGCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMGCFG_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_GMGCFG_CAL_DRVUP
18:14	0x12	CFG2TMC_GMGCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_GMGCFG_LPMD: data pins low power mode select
3	0x0	CFG2TMC_GMGCFG_SCHMT_EN: data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMGCFG_HSM_EN: data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.37 APB_MISC_GP_GMHCFGPADCTRL_0

GMHCFG Pad Control Register

Offset: 91ch | Read/Write: R/W | Reset: 0b1111xxxx1011010010xxxxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_GMHCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GMHCFG_CAL_DRVDN_SLWR
23:19	0x16	CFG2TMC_GMHCFG_CAL_DRVUP
18:14	0x12	CFG2TMC_GMHCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_GMHCFG_LPMD: data pins low power mode select
3	0x0	CFG2TMC_GMHCFG_SCHMT_EN: data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GMHCFG_HSM_EN: data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.38 APB_MISC_GP_OWRCFGPADCTRL_0

OWRCFG Pad Control Register

Offset: 920h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_OWRCFG_CAL_DRVUP_SLWF

Bit	Reset	Description
29:28	0x3	CFG2TMC_OWRCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_OWRCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_OWRCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_OWRCFG_LPMD: data pins low power mode select
3	0x0	CFG2TMC_OWRCFG_SCHMT_EN: data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_OWRCFG_HSM_EN: data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.39 APB_MISC_GP_UADCFGPADCTRL_0

UDACFG Pad Control Register

Offset: 924h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_UDACFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_UDACFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_UDACFG_CAL_DRVUP
16:12	0x12	CFG2TMC_UDACFG_CAL_DRVDN
5:4	0x3	CFG2TMC_UDACFG_LPMD: data pins low power mode select
3	0x0	CFG2TMC_UDACFG_SCHMT_EN: data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_UDACFG_HSM_EN: data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.40 APB_MISC_GP_GPVCFGPADCTRL_0

GPV Pad Control Register

Offset: 928h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_GPVCFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_GPVCFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_GPVCFG_CAL_DRVUP
16:12	0x12	CFG2TMC_GPVCFG_CAL_DRVDN
5:4	0x3	CFG2TMC_GPVCFG_LPMD: data pins low power mode select
3	0x0	CFG2TMC_GPVCFG_SCHMT_EN: data pins schmidt enable

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_GPVCFG_HSM_EN: data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.41 APB_MISC_GP_DEV3CFGPADCTRL_0

DEV3CFG Pad Control Register

Offset: 92ch | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

Bit	Reset	Description
31:30	0x3	CFG2TMC_DEV3CFG_CAL_DRVUP_SLWF
29:28	0x3	CFG2TMC_DEV3CFG_CAL_DRVDN_SLWR
24:20	0x16	CFG2TMC_DEV3CFG_CAL_DRVUP
16:12	0x12	CFG2TMC_DEV3CFG_CAL_DRVDN
5:4	0x3	CFG2TMC_DEV3CFG_LPMD: DEV3CFG data pins low power mode select
3	0x0	CFG2TMC_DEV3CFG_SCHMT_EN: DEV3CFG data pins schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_DEV3CFG_HSM_EN: DEV3CFG data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.4.42 APB_MISC_GP_SDMMC4_VTTGENPADCTRL_0

SDMMC4 GMI Pins VTTGEN Pad Control Register

Offset: 930h | Read/Write: R/W | Reset: 0b000xxxxx000x101x101xxxxx0x0

Bit	Reset	Description
26:24	0x0	CFG2TMC_SDMMC4_VTTGEN_DRVUP: [PMC]
18:16	0x0	CFG2TMC_SDMMC4_VTTGEN_DRVDN: [PMC]
14:12	0x5	CFG2TMC_SDMMC4_VTTGEN_VAUXP_LEVEL: [PMC]
10:8	0x5	CFG2TMC_SDMMC4_VTTGEN_VCLAMP_LEVEL: [PMC]
2	0x0	CFG2TMC_SDMMC4_VTTGEN_E_DDR3: [PMC] For SDMMC this Bit Shouldnt be SET.
1	X	CFG2TMC_SDMMC4_VTTGEN_SHORT_PWRGND: [PMC] dummy pin
0	0x0	CFG2TMC_SDMMC4_VTTGEN_SHORT: [PMC]

20.1.4.43 APB_MISC_GP_NONDDR_LPDDR_PADCTRL_0

These pads are just used to support 1.8V/1.2V. The DDR functionality of the pads is not used.

NON DDR LPDDR Pads e_bypass/e_gpio/io_reset_n Control

Offset: 934h | Read/Write: R/W | Reset: 0b01xxxxxx01

Bit	Reset	Description
9	0x0	NONDDR_LPDDR_GMI_E_BYPASS
8	0x1	NONDDR_LPDDR_GMI_E_GPIO
1	0x0	NONDDR_LPDDR_DVI_E_BYPASS
0	0x1	NONDDR_LPDDR_DVI_E_GPIO

20.1.4.44 APB_MISC_GP_CECCFGPADCTRL_0

CECCFG Pad Control Register

Offset: 938h | Read/Write: R/W | Reset: 0b1111xxx10110xxx10010xxxxxx1100

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
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 schmidt enable 0 = DISABLE 1 = ENABLE
2	0x0	CFG2TMC_CECCFG_HSM_EN: data pins high speed mode enable 0 = DISABLE 1 = ENABLE

20.1.5 SATA AUX Registers

20.1.5.1 SATA_AUX_MISC_CNTL_1_0

MISC_CNTL_1 register

Offset: 1108h | Read/Write: R/W | Reset: 0bxx00xx00000

Bit	R/W	Reset	Description
10:9	RO	X	SATA2IPSM_ST: Indicate whether the Sata link is in partial/slumber modes. Used for debug purpose.
8	RW	0x0	NVA2SATA_OOB_ON_SCONTROL_SPD_WR: If OOB_ON_SCONTROL_SPD_WR is set to 1 then sata controller will do OOB and go through speed negotiation with the drive whenever its SCONTROL_SPD register is written. 0 = NO 1 = YES
7	RW	0x0	NVA2SATA_OOB_ON_POR: Controls whether SATA controllers do OOB sequence automatically when they come out of reset or not. 0 = NO 1 = YES
6:5	RO	X	L0_RX_IDLE_T_SAX: l0_rx_idle_t value from Sata controller When SAX partition is power-gated, this field will show clamped value 0 = NORMAL
4:3	RW	0x0	L0_RX_IDLE_T_NPG: set l0_rx_idle_t value for Sata phy from apb_misc(non-power-gated logic) Before SAX partition is power-gated, SATA_l0_rx_idle_t_npg needs to have the same value as SATA_l0_rx_idle_t_sax then set SATA_l0_rx_idle_t_mux to '1'. This is to make sure Sata PHY is still receiving correct threadhold setting for OOB detection when SAX is power-gated. After SAX power is restored, same value will be restored back to l0_rx_idle_t register in Sata controller then SATA_l0_rx_idle_t_mux can be set back to '0'. 0 = NORMAL
2	RW	0x0	L0_RX_IDLE_T_MUX: select l0_rx_idle_t driving source for sata phy 0 : from sata controller 1 : from apb_misc 0 = FROM_SATA 1 = FROM_APB_MISC
1	RW	0x0	PMU2SATA_ACCLMTR_TRIG: This config bit is used inside SATA to select between the two accelerometer triggers, between External trigger & PMU's trigger. 0 : external trigger disable 1 : external trigger enable 0 = DISABLE 1 = ENABLE
0	RW	0x0	DEVICE_DIS_SATA0: Serial ATA Interface 0 Disable 1 = Internal Serial ATA Interface 0 Disabled (Not seen as part of PCI space) 0 = Internal Serial ATA Interface 0 Enabled. 0 = ENABLE 1 = DISABLE

20.1.5.2 SATA_AUX_RX_STAT_INT_0

RX_STAT_INT register

Offset: 110ch | Read/Write: RO | Reset: 0bxx

Bit	Reset	Description
1	X	SATA_L0_RX_STAT_IDLE: Sata pad L0 rx_stat_idle status 1 : rx path is idle 0 : rx path is active 0 = NO 1 = YES

Bit	Reset	Description
0	X	SATA_RX_STAT_INT_STATUS: Sata rx_stat interrupt status from sata pad 1 = rx_stat interrupt is pending 0 = rx_stat interrupt is not pending 0 = NONE 1 = PENDING

20.1.5.3 SATA_AUX_RX_STAT_SET_0

RX_STAT_SET register

Offset: 1110h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	SATA_RX_STAT_INT_SET: Sata rx_stat interrupt is set (this bit is auto-cleared) set interrupt status bit to '1' when register is written read back value is always '0'

20.1.5.4 SATA_AUX_RX_STAT_CLR_0

RX_STAT_CLR register

Offset: 1114h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	SATA_RX_STAT_INT_CLR: Sata rx_stat interrupt is cleared (this bit is auto-cleared) clear interrupt status bit to '0' when register is written read back value is always '0'

List of input pins to be tied off:

- bst_bond_sata2 - tie to '1'
- storage_feature - tie to '0'
- dfa2fpci_cmd_compat - tie to '0', ipfs doesn't have this output
- dfa2fpci_cmd_passpw - tie to '0', ipfs doesn't have this output
- ufa2fpci_arb_block_coherent_np - tie to '0', ipfs doesn't have this output
- ufa2fpci_arb_block_coherent_pw - tie to '0', ipfs doesn't have this output
- pri_page_offset_endpoint_cfg - tie to 32'h50100000
- leg2dev_devnum_sata0 - tie to 5'h0A

Register bits that are not needed:

- 1b sata2pmu_wake R
- 4b fpci2pmu_core_act_sts_sata0 R
- 1b fpci2pmu_core_act_tog_sata0 R
- 16b sata2sm_pg_info R
- 1b sata2sm_pll_powerdown R
- 1b sata2smi_bohc_ooc_change R

20.1.6 Pin Mux Selects

This section defines the Pin mux selects, Pullup Pulldn and E_input programmability for each Pin/Ball in Tegra 3. Every register has 7 bits defined which are used for Functional Pin Select. The rest are specific to few pads like OD, IO_RESET.

1:0 PM --> Functional Pin Select.

3:2 PUPD --> Pullup / PullDn control.

4:4 Tristate --> Tristate Enable / Disable.

5:5 E_INPUT --> Input Receiver Enable/Disable.

7:7 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)

8:8 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 de-asserted.

--> If IO_RESET_N is de-asserted before core controls come up, there could be unknown outputs or glitches.

Optional bits

6:6 OD --> Open Drain Enable / Disable. Specifically for I2C pads.

20.1.6.1 PINMUX_AUX_ULPI_DATA0_0

Offset: 3000h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPI3 1 = HSI 2 = UARTA 3 = ULPI

20.1.6.2 PINMUX_AUX_ULPI_DATA1_0

Offset: 3004h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPI3 1 = HSI 2 = UARTA 3 = ULPI

20.1.6.3 PINMUX_AUX_ULPI_DATA2_0

Offset: 3008h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPI3 1 = HSI 2 = UARTA 3 = ULPI

20.1.6.4 PINMUX_AUX_ULPI_DATA3_0

Offset: 300ch | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPI3 1 = HSI 2 = UARTA 3 = ULPI

20.1.6.5 PINMUX_AUX_ULPI_DATA4_0

Offset: 3010h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPI2 1 = HSI 2 = UARTA 3 = ULPI

20.1.6.6 PINMUX_AUX_ULPI_DATA5_0

Offset: 3014h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPI2 1 = HSI 2 = UARTA 3 = ULPI

20.1.6.7 PINMUX_AUX_ULPI_DATA6_0

Offset: 3018h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPI2 1 = HSI 2 = UARTA 3 = ULPI

20.1.6.8 PINMUX_AUX_ULPI_DATA7_0

Offset: 301ch | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPI2 1 = HSI 2 = UARTA 3 = ULPI

20.1.6.9 PINMUX_AUX_ULPI_CLK_0

Offset: 3020h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = SPI1 1 = RSVD 2 = UARTD 3 = ULPI

20.1.6.10 PINMUX_AUX_ULPI_DIR_0

Offset: 3024h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = SPI1 1 = RSVD 2 = UARTD 3 = ULPI

20.1.6.11 PINMUX_AUX_ULPI_NXT_0

Offset: 3028h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = SPI1 1 = RSVD 2 = UARTD 3 = ULPI

20.1.6.12 PINMUX_AUX_ULPI_STP_0

Offset: 302ch | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = SPI1 1 = RSVD 2 = UARTD 3 = ULPI

20.1.6.13 PINMUX_AUX_DAP3_FS_0

Offset: 3030h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S2 1 = RSVD1 2 = DISPLAYA 3 = DISPLAYB

20.1.6.14 PINMUX_AUX_DAP3_DIN_0

Offset: 3034h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S2 1 = RSVD1 2 = DISPLAYA 3 = DISPLAYB

20.1.6.15 PINMUX_AUX_DAP3_DOUT_0

Offset: 3038h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S2 1 = RSVD1 2 = DISPLAYA 3 = DISPLAYB

20.1.6.16 PINMUX_AUX_DAP3_SCLK_0

Offset: 303ch | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S2 1 = RSVD1 2 = DISPLAYA 3 = DISPLAYB

20.1.6.17 PINMUX_AUX_GPIO_PV0_0

Offset: 3040h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM

20.1.6.18 PINMUX_AUX_GPIO_PV1_0

Offset: 3044h | Read/Write: R/W | Reset: 0b0x110000

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	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM

20.1.6.19 PINMUX_AUX_SDMMC1_CLK_0

Offset: 3048h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SDMMC1 1 = RSVD1 2 = RSVD2 3 = UARTA

20.1.6.20 PINMUX_AUX_SDMMC1_CMD_0

Offset: 304ch | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SDMMC1 1 = RSVD1 2 = RSVD2 3 = UARTA

20.1.6.21 PINMUX_AUX_SDMMC1_DAT3_0

Offset: 3050h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SDMMC1 1 = RSVD1 2 = UARTE 3 = UARTA

20.1.6.22 PINMUX_AUX_SDMMC1_DAT2_0

Offset: 3054h | Read/Write: R/W | Reset: 0b0x111000

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	ENABLE	E_INPUT: 0 = DISABLE 1 = ENABLE
4	TRISTATE	TRISTATE: 0 = NORMAL

Bit	Reset	Description
		1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM: 0 = SDMMC1 1 = RSVD1 2 = UARTE 3 = UARTA

20.1.6.23 PINMUX_AUX_SDMMC1_DAT1_0

Offset: 3058h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SDMMC1 1 = RSVD1 2 = UARTE 3 = UARTA

20.1.6.24 PINMUX_AUX_SDMMC1_DAT0_0

Offset: 305ch | Read/Write: R/W | Reset: 0b0x111000

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

Bit	Reset	Description
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM: 0 = SDMMC1 1 = RSVD1 2 = UARTE 3 = UARTA

20.1.6.25 PINMUX_AUX_GPIO_PV2_0

Offset: 3060h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = OWR 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.26 PINMUX_AUX_GPIO_PV3_0

Offset: 3064h | Read/Write: R/W | Reset: 0b0x110000

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

Bit	Reset	Description
		1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM: 0 = CLK12 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.27 PINMUX_AUX_CLK2_OUT_0

Offset: 3068h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = EXTPERIPH2 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.28 PINMUX_AUX_CLK2_REQ_0

Offset: 306ch | Read/Write: R/W | Reset: 0b0x110000

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

Bit	Reset	Description
		3 = RSVD
1:0	0x0	PM: 0 = DAP 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.29 PINMUX_AUX_LCD_PWR1_0

Offset: 3070h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.30 PINMUX_AUX_LCD_PWR2_0

Offset: 3074h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = SPI5 3 = HDMI

20.1.6.31 PINMUX_AUX_LCD_SDIN_0

Offset: 3078h | Read/Write: R/W | Reset: 0b0x011000

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = SPI5 3 = RSVD

20.1.6.32 PINMUX_AUX_LCD_SDOUT_0

Offset: 307ch | Read/Write: R/W | Reset: 0b0x011000

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA

Bit	Reset	Description
		1 = DISPLAYB 2 = SPI5 3 = HDMI

20.1.6.33 PINMUX_AUX_LCD_WR_N_0

Offset: 3080h | Read/Write: R/W | Reset: 0b0x011000

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = SPI5 3 = HDMI

20.1.6.34 PINMUX_AUX_LCD_CS0_N_0

Offset: 3084h | Read/Write: R/W | Reset: 0b0x011000

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = SPI5

Bit	Reset	Description
		3 = RSVD

20.1.6.35 PINMUX_AUX_LCD_DC0_0

Offset: 3088h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.36 PINMUX_AUX_LCD_SCK_0

Offset: 308ch | Read/Write: R/W | Reset: 0b0x011000

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = SPI5 3 = HDMI

20.1.6.37 PINMUX_AUX_LCD_PWR0_0

Offset: 3090h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = SPI5 3 = HDMI

20.1.6.38 PINMUX_AUX_LCD_PCLK_0

Offset: 3094h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.39 PINMUX_AUX_LCD_DE_0

Offset: 3098h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.40 PINMUX_AUX_LCD_HSYNC_0

Offset: 309ch | Read/Write: R/W | Reset: 0b0x011000

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.41 PINMUX_AUX_LCD_VSYNC_0

Offset: 30a0h | Read/Write: R/W | Reset: 0b0x011000

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.42 PINMUX_AUX_LCD_D0_0

Offset: 30a4h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.43 PINMUX_AUX_LCD_D1_0

Offset: 30a8h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.44 PINMUX_AUX_LCD_D2_0

Offset: 30ach | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.45 PINMUX_AUX_LCD_D3_0

Offset: 30b0h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.46 PINMUX_AUX_LCD_D4_0

Offset: 30b4h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.47 PINMUX_AUX_LCD_D5_0

Offset: 30b8h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.48 PINMUX_AUX_LCD_D6_0

Offset: 30bch | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.49 PINMUX_AUX_LCD_D7_0

Offset: 30c0h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.50 PINMUX_AUX_LCD_D8_0

Offset: 30c4h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.51 PINMUX_AUX_LCD_D9_0

Offset: 30c8h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.52 PINMUX_AUX_LCD_D10_0

Offset: 30cch | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.53 PINMUX_AUX_LCD_D11_0

Offset: 30d0h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.54 PINMUX_AUX_LCD_D12_0

Offset: 30d4h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.55 PINMUX_AUX_LCD_D13_0

Offset: 30d8h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.56 PINMUX_AUX_LCD_D14_0

Offset: 30dch | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.57 PINMUX_AUX_LCD_D15_0

Offset: 30e0h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.58 PINMUX_AUX_LCD_D16_0

Offset: 30e4h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.59 PINMUX_AUX_LCD_D17_0

Offset: 30e8h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.60 PINMUX_AUX_LCD_D18_0

Offset: 30ech | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.61 PINMUX_AUX_LCD_D19_0

Offset: 30f0h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.62 PINMUX_AUX_LCD_D20_0

Offset: 30f4h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.63 PINMUX_AUX_LCD_D21_0

Offset: 30f8h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.64 PINMUX_AUX_LCD_D22_0

Offset: 30fch | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.65 PINMUX_AUX_LCD_D23_0

Offset: 3100h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.66 PINMUX_AUX_LCD_CS1_N_0

Offset: 3104h | Read/Write: R/W | Reset: 0b0x011000

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = SPI5 3 = RSVD2

20.1.6.67 PINMUX_AUX_LCD_M1_0

Offset: 3108h | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.68 PINMUX_AUX_LCD_DC1_0

Offset: 310ch | Read/Write: R/W | Reset: 0b0x010100

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE
5	DISABLE	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	0x0	PM: 0 = DISPLAYA 1 = DISPLAYB 2 = RSVD1 3 = RSVD2

20.1.6.69 PINMUX_AUX_HDMI_INT_0

Offset: 3110h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM

20.1.6.70 PINMUX_AUX_DDC_SCL_0

Offset: 3114h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = I2C4 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.71 PINMUX_AUX_DDC_SDA_0

Offset: 3118h | Read/Write: R/W | Reset: 0b0x110000

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	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM: 0 = I2C4 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.72 PINMUX_AUX_CRT_HSYNC_0

Offset: 311ch | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = CRT 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.73 PINMUX_AUX_CRT_VSYNC_0

Offset: 3120h | Read/Write: R/W | Reset: 0b0x111000

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_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM: 0 = CRT 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.74 PINMUX_AUX_VI_D0_0

Offset: 3124h | Read/Write: R/W | Reset: 0b10x110100

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
1:0	0x0	PM: 0 = DDR 1 = RSVD1 2 = VI 3 = RSVD2

20.1.6.75 PINMUX_AUX_VI_D1_0

Offset: 3128h | Read/Write: R/W | Reset: 0b10x110100

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL

Bit	Reset	Description
		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
1:0	0x0	PM: 0 = DDR 1 = SDMMC2 2 = VI 3 = RSVD1

20.1.6.76 PINMUX_AUX_VI_D2_0

Offset: 312ch | Read/Write: R/W | Reset: 0b10x110100

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
1:0	0x0	PM: 0 = DDR 1 = SDMMC2 2 = VI 3 = RSVD1

20.1.6.77 PINMUX_AUX_VI_D3_0

Offset: 3130h | Read/Write: R/W | Reset: 0b10x110100

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
1:0	0x0	PM: 0 = DDR 1 = SDMMC2 2 = VI 3 = RSVD1

20.1.6.78 PINMUX_AUX_VI_D4_0

Offset: 3134h | Read/Write: R/W | Reset: 0b10x110100

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
1:0	0x0	PM: 0 = DDR 1 = SDMMC2 2 = VI

Bit	Reset	Description
		3 = RSVD1

20.1.6.79 PINMUX_AUX_VI_D5_0

Offset: 3138h | Read/Write: R/W | Reset: 0b10x110100

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
1:0	0x0	PM: 0 = DDR 1 = SDMMC2 2 = VI 3 = RSVD1

20.1.6.80 PINMUX_AUX_VI_D6_0

Offset: 313ch | Read/Write: R/W | Reset: 0b10x110100

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

Bit	Reset	Description
		3 = RSVD
1:0	0x0	PM: 0 = DDR 1 = SDMMC2 2 = VI 3 = RSVD1

20.1.6.81 PINMUX_AUX_VI_D7_0

Offset: 3140h | Read/Write: R/W | Reset: 0b10x110100

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
1:0	0x0	PM: 0 = DDR 1 = SDMMC2 2 = VI 3 = RSVD1

20.1.6.82 PINMUX_AUX_VI_D8_0

Offset: 3144h | Read/Write: R/W | Reset: 0b10x110100

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

Bit	Reset	Description
		1 = TRISTATE
3:2	PULL_DOWN	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM: 0 = DDR 1 = SDMMC2 2 = VI 3 = RSVD1

20.1.6.83 PINMUX_AUX_VI_D9_0

Offset: 3148h | Read/Write: R/W | Reset: 0b10x110100

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
1:0	0x0	PM: 0 = DDR 1 = SDMMC2 2 = VI 3 = RSVD1

20.1.6.84 PINMUX_AUX_VI_D10_0

Offset: 314ch | Read/Write: R/W | Reset: 0b10x110100

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
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	0x0	PM: 0 = DDR 1 = RSVD1 2 = VI 3 = RSVD2

20.1.6.85 PINMUX_AUX_VI_D11_0

Offset: 3150h | Read/Write: R/W | Reset: 0b10x110100

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
1:0	0x0	PM: 0 = DDR 1 = RSVD1 2 = VI 3 = RSVD2

20.1.6.86 PINMUX_AUX_VI_PCLK_0

Offset: 3154h | Read/Write: R/W | Reset: 0b10x110100

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL

Bit	Reset	Description
		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
1:0	0x0	PM: 0 = RSVD1 1 = SDMMC2 2 = VI 3 = RSVD2

20.1.6.87 PINMUX_AUX_VI_MCLK_0

Offset: 3158h | Read/Write: R/W | Reset: 0b10x110100

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
1:0	0x0	PM: 0 = VI 1 = VI_ALT1 2 = VI_ALT2 3 = VI_ALT3

20.1.6.88 PINMUX_AUX_VI_VSYNC_0

Offset: 315ch | Read/Write: R/W | Reset: 0b10x110100

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
1:0	0x0	PM: 0 = DDR 1 = RSVD1 2 = VI 3 = RSVD2

20.1.6.89 PINMUX_AUX_VI_HSYNC_0

Offset: 3160h | Read/Write: R/W | Reset: 0b10x110100

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
1:0	0x0	PM: 0 = DDR 1 = RSVD1 2 = VI

Bit	Reset	Description
		3 = RSVD2

20.1.6.90 PINMUX_AUX_UART2_RXD_0

Offset: 3164h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = IRDA 1 = SPDIF 2 = UARTA 3 = SPI4

20.1.6.91 PINMUX_AUX_UART2_TXD_0

Offset: 3168h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = IRDA 1 = SPDIF 2 = UARTA 3 = SPI4

20.1.6.92 PINMUX_AUX_UART2_RTS_N_0

Offset: 316ch | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = UARTA 1 = UARTB 2 = GMI 3 = SPI4

20.1.6.93 PINMUX_AUX_UART2_CTS_N_0

Offset: 3170h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = UARTA 1 = UARTB 2 = GMI 3 = SPI4

20.1.6.94 PINMUX_AUX_UART3_TXD_0

Offset: 3174h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = UARTC 1 = RSVD1 2 = GMI 3 = RSVD2

20.1.6.95 PINMUX_AUX_UART3_RXD_0

Offset: 3178h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = UARTC 1 = RSVD1 2 = GMI 3 = RSVD2

20.1.6.96 PINMUX_AUX_UART3_CTS_N_0

Offset: 317ch | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = UARTC 1 = RSVD1 2 = GMI 3 = RSVD2

20.1.6.97 PINMUX_AUX_UART3_RTS_N_0

Offset: 3180h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = UARTC 1 = PWM0 2 = GMI 3 = RSVD2

20.1.6.98 PINMUX_AUX_GPIO_PU0_0

Offset: 3184h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = OWR 1 = UARTA 2 = GMI 3 = RSVD1

20.1.6.99 PINMUX_AUX_GPIO_PU1_0

Offset: 3188h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = RSVD1 1 = UARTA 2 = GMI 3 = RSVD2

20.1.6.100 PINMUX_AUX_GPIO_PU2_0

Offset: 318ch | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = RSVD1 1 = UARTA 2 = GMI 3 = RSVD2

20.1.6.101 PINMUX_AUX_GPIO_PU3_0

Offset: 3190h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PWM0 1 = UARTA 2 = GMI 3 = RSVD1

20.1.6.102 PINMUX_AUX_GPIO_PU4_0

Offset: 3194h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PWM1 1 = UARTA 2 = GMI 3 = RSVD1

20.1.6.103 PINMUX_AUX_GPIO_PU5_0

Offset: 3198h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PWM2 1 = UARTA 2 = GMI 3 = RSVD1

20.1.6.104 PINMUX_AUX_GPIO_PU6_0

Offset: 319ch | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PWM3 1 = UARTA 2 = GMI 3 = RSVD1

20.1.6.105 PINMUX_AUX_GEN1_I2C_SDA_0

Offset: 31a0h | Read/Write: R/W | Reset: 0b01100000

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	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM: 0 = I2C1 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.106 PINMUX_AUX_GEN1_I2C_SCL_0

Offset: 31a4h | Read/Write: R/W | Reset: 0b01100000

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	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM: 0 = I2C1 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.107 PINMUX_AUX_DAP4_FS_0

Offset: 31a8h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S3 1 = RSVD1 2 = GMI 3 = RSVD2

20.1.6.108 PINMUX_AUX_DAP4_DIN_0

Offset: 31ach | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S3 1 = RSVD1 2 = GMI 3 = RSVD2

20.1.6.109 PINMUX_AUX_DAP4_DOUT_0

Offset: 31b0h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S3 1 = RSVD1 2 = GMI 3 = RSVD2

20.1.6.110 PINMUX_AUX_DAP4_SCLK_0

Offset: 31b4h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S3 1 = RSVD1 2 = GMI 3 = RSVD2

20.1.6.111 PINMUX_AUX_CLK3_OUT_0

Offset: 31b8h | Read/Write: R/W | Reset: 0b0x100000

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	0x0	PM: 0 = EXTPERIPH3 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.112 PINMUX_AUX_CLK3_REQ_0

Offset: 31bch | Read/Write: R/W | Reset: 0b0x100000

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	0x0	PM: 0 = DEV3 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.113 PINMUX_AUX_GMI_WP_N_0

Offset: 31c0h | Read/Write: R/W | Reset: 0b0x111010

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 = RSVD1 1 = NAND 2 = GMI 3 = GMI_ALT

20.1.6.114 PINMUX_AUX_GMI_IORDY_0

Offset: 31c4h | Read/Write: R/W | Reset: 0b0x101010

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	GMI	PM: 0 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.115 PINMUX_AUX_GMI_WAIT_0

Offset: 31c8h | Read/Write: R/W | Reset: 0b0x101010

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	GMI	PM: 0 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.116 PINMUX_AUX_GMI_ADV_N_0

Offset: 31cch | Read/Write: R/W | Reset: 0b0x100010

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	GMI	PM: 0 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.117 PINMUX_AUX_GMI_CLK_0

Offset: 31d0h | Read/Write: R/W | Reset: 0b0x100010

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	GMI	PM: 0 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.118 PINMUX_AUX_GMI_CS0_N_0

Offset: 31d4h | Read/Write: R/W | Reset: 0b0x111010

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 = RSVD1 1 = NAND 2 = GMI 3 = DTV

20.1.6.119 PINMUX_AUX_GMI_CS1_N_0

Offset: 31d8h | Read/Write: R/W | Reset: 0b0x111010

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 = RSVD1 1 = NAND 2 = GMI 3 = DTV

20.1.6.120 PINMUX_AUX_GMI_CS2_N_0

Offset: 31dch | Read/Write: R/W | Reset: 0b0x101010

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	GMI	PM: 0 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.121 PINMUX_AUX_GMI_CS3_N_0

Offset: 31e0h | Read/Write: R/W | Reset: 0b0x101010

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	GMI	PM: 0 = RSVD1 1 = NAND 2 = GMI 3 = GMI_ALT

20.1.6.122 PINMUX_AUX_GMI_CS4_N_0

Offset: 31e4h | Read/Write: R/W | Reset: 0b0x111010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.123 PINMUX_AUX_GMI_CS6_N_0

Offset: 31e8h | Read/Write: R/W | Reset: 0b0x101010

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	GMI	PM: 0 = NAND 1 = NAND_ALT 2 = GMI 3 = SATA

20.1.6.124 PINMUX_AUX_GMI_CS7_N_0

Offset: 31ech | Read/Write: R/W | Reset: 0b0x111010

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 = GMI_ALT

20.1.6.125 PINMUX_AUX_GMI_AD0_0

Offset: 31f0h | Read/Write: R/W | Reset: 0b0x110010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.126 PINMUX_AUX_GMI_AD1_0

Offset: 31f4h | Read/Write: R/W | Reset: 0b0x110010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.127 PINMUX_AUX_GMI_AD2_0

Offset: 31f8h | Read/Write: R/W | Reset: 0b0x110010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.128 PINMUX_AUX_GMI_AD3_0

Offset: 31fch | Read/Write: R/W | Reset: 0b0x110010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.129 PINMUX_AUX_GMI_AD4_0

Offset: 3200h | Read/Write: R/W | Reset: 0b0x110010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.130 PINMUX_AUX_GMI_AD5_0

Offset: 3204h | Read/Write: R/W | Reset: 0b0x110010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.131 PINMUX_AUX_GMI_AD6_0

Offset: 3208h | Read/Write: R/W | Reset: 0b0x110010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.132 PINMUX_AUX_GMI_AD7_0

Offset: 320ch | Read/Write: R/W | Reset: 0b0x110010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.133 PINMUX_AUX_GMI_AD8_0

Offset: 3210h | Read/Write: R/W | Reset: 0b0x110110

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 = RSVD2

20.1.6.134 PINMUX_AUX_GMI_AD9_0

Offset: 3214h | Read/Write: R/W | Reset: 0b0x110110

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 = RSVD2

20.1.6.135 PINMUX_AUX_GMI_AD10_0

Offset: 3218h | Read/Write: R/W | Reset: 0b0x110110

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 = PWM2 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.136 PINMUX_AUX_GMI_AD11_0

Offset: 321ch | Read/Write: R/W | Reset: 0b0x110110

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 = RSVD2

20.1.6.137 PINMUX_AUX_GMI_AD12_0

Offset: 3220h | Read/Write: R/W | Reset: 0b0x110010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.138 PINMUX_AUX_GMI_AD13_0

Offset: 3224h | Read/Write: R/W | Reset: 0b0x110010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.139 PINMUX_AUX_GMI_AD14_0

Offset: 3228h | Read/Write: R/W | Reset: 0b0x110010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.140 PINMUX_AUX_GMI_AD15_0

Offset: 322ch | Read/Write: R/W | Reset: 0b0x110010

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 = RSVD1 1 = NAND 2 = GMI 3 = RSVD2

20.1.6.141 PINMUX_AUX_GMI_A16_0

Offset: 3230h | Read/Write: R/W | Reset: 0b0x110010

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 2 = GMI 3 = GMI_ALT

20.1.6.142 PINMUX_AUX_GMI_A17_0

Offset: 3234h | Read/Write: R/W | Reset: 0b0x110010

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 2 = GMI 3 = DTV

20.1.6.143 PINMUX_AUX_GMI_A18_0

Offset: 3238h | Read/Write: R/W | Reset: 0b0x110010

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 2 = GMI 3 = DTV

20.1.6.144 PINMUX_AUX_GMI_A19_0

Offset: 323ch | Read/Write: R/W | Reset: 0b0x110010

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 2 = GMI 3 = RSVD3

20.1.6.145 PINMUX_AUX_GMI_WR_N_0

Offset: 3240h | Read/Write: R/W | Reset: 0b0x100010

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	GMI	PM: 0 = RSVD1 1 = NAND 2 = GMI 3 = RSVD3

20.1.6.146 PINMUX_AUX_GMI_OE_N_0

Offset: 3244h | Read/Write: R/W | Reset: 0b0x100010

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	GMI	PM: 0 = RSVD1 1 = NAND 2 = GMI 3 = RSVD3

20.1.6.147 PINMUX_AUX_GMI_DQS_0

Offset: 3248h | Read/Write: R/W | Reset: 0b0x100000

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	RSVD1	PM: 0 = RSVD1 1 = NAND 2 = GMI 3 = RSVD3

20.1.6.148 PINMUX_AUX_GMI_RST_N_0

Offset: 324ch | Read/Write: R/W | Reset: 0b0x101010

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	GMI	PM: 0 = NAND 1 = NAND_ALT 2 = GMI 3 = RSVD3

20.1.6.149 PINMUX_AUX_GEN2_I2C_SCL_0

Offset: 3250h | Read/Write: R/W | Reset: 0b01110000

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 = HDMI 2 = GMI 3 = RSVD3

20.1.6.150 PINMUX_AUX_GEN2_I2C_SDA_0

Offset: 3254h | Read/Write: R/W | Reset: 0b01110000

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 = HDMI 2 = GMI 3 = RSVD3

20.1.6.151 PINMUX_AUX_SDMMC4_CLK_0

Offset: 3258h | Read/Write: R/W | Reset: 0b10x101000

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	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM: 0 = VI_ALT3 1 = NAND 2 = GMI

Bit	Reset	Description
		3 = SDMMC4

20.1.6.152 PINMUX_AUX_SDMMC4_CMD_0

Offset: 325ch | Read/Write: R/W | Reset: 0b10x111000

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	0x0	PM: 0 = I2C3 1 = NAND 2 = GMI 3 = SDMMC4

20.1.6.153 PINMUX_AUX_SDMMC4_DAT0_0

Offset: 3260h | Read/Write: R/W | Reset: 0b10x100010

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	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP

Bit	Reset	Description
		3 = RSVD
1:0	GMI	PM: 0 = UARTE 1 = SPI3 2 = GMI 3 = SDMMC4

20.1.6.154 PINMUX_AUX_SDMMC4_DAT1_0

Offset: 3264h | Read/Write: R/W | Reset: 0b10x100010

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	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 = UARTE 1 = SPI3 2 = GMI 3 = SDMMC4

20.1.6.155 PINMUX_AUX_SDMMC4_DAT2_0

Offset: 3268h | Read/Write: R/W | Reset: 0b10x100010

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	NORMAL	TRISTATE: 0 = NORMAL

Bit	Reset	Description
		1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	GMI	PM: 0 = UARTE 1 = SPI3 2 = GMI 3 = SDMMC4

20.1.6.156 PINMUX_AUX_SDMMC4_DAT3_0

Offset: 326ch | Read/Write: R/W | Reset: 0b10x100010

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	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 = UARTE 1 = SPI3 2 = GMI 3 = SDMMC4

20.1.6.157 PINMUX_AUX_SDMMC4_DAT4_0

Offset: 3270h | Read/Write: R/W | Reset: 0b10x100010

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL 1 = IORESET
7	DISABLE	LOCK: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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	GMI	PM: 0 = I2C3 1 = I2S4 2 = GMI 3 = SDMMC4

20.1.6.158 PINMUX_AUX_SDMMC4_DAT5_0

Offset: 3274h | Read/Write: R/W | Reset: 0b10x100010

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	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 = VGP3 1 = I2S4 2 = GMI 3 = SDMMC4

20.1.6.159 PINMUX_AUX_SDMMC4_DAT6_0

Offset: 3278h | Read/Write: R/W | Reset: 0b10x100010

Bit	Reset	Description
8	IORESET	IO_RESET: 0 = NORMAL

Bit	Reset	Description
		1 = IORESET
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	GMI	PM: 0 = VGP4 1 = I2S4 2 = GMI 3 = SDMMC4

20.1.6.160 PINMUX_AUX_SDMMC4_DAT7_0

Offset: 327ch | Read/Write: R/W | Reset: 0b10x100010

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	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 = VGP5 1 = I2S4 2 = GMI 3 = SDMMC4

20.1.6.161 PINMUX_AUX_SDMMC4_RST_N_0

Offset: 3280h | Read/Write: R/W | Reset: 0b10x101011

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	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	PULL_UP	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	POPSDMMC4	PM: 0 = VGP6 1 = RSVD1 2 = RSVD2 3 = POPSDMMC4

20.1.6.162 PINMUX_AUX_CAM_MCLK_0

Offset: 3284h | Read/Write: R/W | Reset: 0b0x111010

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	VI_ALT3	PM: 0 = VI 1 = VI_ALT1 2 = VI_ALT3 3 = POPSDMMC4

20.1.6.163 PINMUX_AUX_GPIO_PCC1_0

Offset: 3288h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = I2S4 1 = RSVD1 2 = RSVD2 3 = POPSDMMC4

20.1.6.164 PINMUX_AUX_GPIO_PBB0_0

Offset: 328ch | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = I2S4 1 = RSVD1 2 = RSVD2 3 = POPSDMMC4

20.1.6.165 PINMUX_AUX_CAM_I2C_SCL_0

Offset: 3290h | Read/Write: R/W | Reset: 0b01110000

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	0x0	PM: 0 = VGP1 1 = I2C3 2 = RSVD2 3 = POPSDMMC4

20.1.6.166 PINMUX_AUX_CAM_I2C_SDA_0

Offset: 3294h | Read/Write: R/W | Reset: 0b01110000

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	0x0	PM: 0 = VGP2 1 = I2C3 2 = RSVD2

Bit	Reset	Description
		3 = POPSDMMC4

20.1.6.167 PINMUX_AUX_GPIO_PBB3_0

Offset: 3298h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = VGP3 1 = DISPLAYA 2 = DISPLAYB 3 = POPSDMMC4

20.1.6.168 PINMUX_AUX_GPIO_PBB4_0

Offset: 329ch | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = VGP4 1 = DISPLAYA 2 = DISPLAYB 3 = POPSDMMC4

20.1.6.169 PINMUX_AUX_GPIO_PBB5_0

Offset: 32a0h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = VGP5 1 = DISPLAYA 2 = DISPLAYB 3 = POPSDMMC4

20.1.6.170 PINMUX_AUX_GPIO_PBB6_0

Offset: 32a4h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = VGP6 1 = DISPLAYA 2 = DISPLAYB 3 = POPSDMMC4

20.1.6.171 PINMUX_AUX_GPIO_PBB7_0

Offset: 32a8h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = I2S4 1 = RSVD1 2 = RSVD2 3 = POPSDMMC4

20.1.6.172 PINMUX_AUX_GPIO_PCC2_0

Offset: 32ach | Read/Write: R/W | Reset: 0b0x101000

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	0x0	PM: 0 = I2S4 1 = RSVD1 2 = RSVD2 3 = POPSDMMC4

20.1.6.173 PINMUX_AUX_JTAG_RTCK_0

Offset: 32b0h | Read/Write: R/W | Reset: 0b0x101000

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	0x0	PM: 0 = RTCK 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.174 PINMUX_AUX_PWR_I2C_SCL_0

Offset: 32b4h | Read/Write: R/W | Reset: 0b01100000

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	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM: 0 = I2CPWR 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.175 PINMUX_AUX_PWR_I2C_SDA_0

Offset: 32b8h | Read/Write: R/W | Reset: 0b01100000

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	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM: 0 = I2CPWR 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.176 PINMUX_AUX_KB_ROW0_0

Offset: 32bch | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = RSVD2 3 = RSVD3

20.1.6.177 PINMUX_AUX_KB_ROW1_0

Offset: 32c0h | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = RSVD2 3 = RSVD3

20.1.6.178 PINMUX_AUX_KB_ROW2_0

Offset: 32c4h | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = RSVD2 3 = RSVD3

20.1.6.179 PINMUX_AUX_KB_ROW3_0

Offset: 32c8h | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = RSVD2 3 = RSVD3

20.1.6.180 PINMUX_AUX_KB_ROW4_0

Offset: 32cch | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = TRACE 3 = RSVD3

20.1.6.181 PINMUX_AUX_KB_ROW5_0

Offset: 32d0h | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = TRACE 3 = OWR

20.1.6.182 PINMUX_AUX_KB_ROW6_0

Offset: 32d4h | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = SDMMC2 3 = MIO

20.1.6.183 PINMUX_AUX_KB_ROW7_0

Offset: 32d8h | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = SDMMC2 3 = MIO

20.1.6.184 PINMUX_AUX_KB_ROW8_0

Offset: 32dch | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = SDMMC2 3 = MIO

20.1.6.185 PINMUX_AUX_KB_ROW9_0

Offset: 32e0h | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = SDMMC2 3 = MIO

20.1.6.186 PINMUX_AUX_KB_ROW10_0

Offset: 32e4h | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = SDMMC2 3 = MIO

20.1.6.187 PINMUX_AUX_KB_ROW11_0

Offset: 32e8h | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = SDMMC2 3 = MIO

20.1.6.188 PINMUX_AUX_KB_ROW12_0

Offset: 32ech | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = SDMMC2 3 = MIO

20.1.6.189 PINMUX_AUX_KB_ROW13_0

Offset: 32f0h | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = SDMMC2 3 = MIO

20.1.6.190 PINMUX_AUX_KB_ROW14_0

Offset: 32f4h | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = SDMMC2 3 = MIO

20.1.6.191 PINMUX_AUX_KB_ROW15_0

Offset: 32f8h | Read/Write: R/W | Reset: 0b0x100100

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	0x0	PM: 0 = KBC 1 = NAND 2 = SDMMC2 3 = MIO

20.1.6.192 PINMUX_AUX_KB_COL0_0

Offset: 32fch | Read/Write: R/W | Reset: 0b0x101000

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	0x0	PM: 0 = KBC 1 = NAND 2 = TRACE 3 = EMC_DLL

20.1.6.193 PINMUX_AUX_KB_COL1_0

Offset: 3300h | Read/Write: R/W | Reset: 0b0x101000

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	0x0	PM: 0 = KBC 1 = NAND 2 = TRACE 3 = EMC_DLL

20.1.6.194 PINMUX_AUX_KB_COL2_0

Offset: 3304h | Read/Write: R/W | Reset: 0b0x101000

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	0x0	PM: 0 = KBC 1 = NAND 2 = TRACE 3 = RSVD

20.1.6.195 PINMUX_AUX_KB_COL3_0

Offset: 3308h | Read/Write: R/W | Reset: 0b0x101000

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	0x0	PM: 0 = KBC 1 = NAND 2 = TRACE 3 = RSVD

20.1.6.196 PINMUX_AUX_KB_COL4_0

Offset: 330ch | Read/Write: R/W | Reset: 0b0x101000

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	0x0	PM: 0 = KBC 1 = NAND 2 = TRACE 3 = RSVD

20.1.6.197 PINMUX_AUX_KB_COL5_0

Offset: 3310h | Read/Write: R/W | Reset: 0b0x101000

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	0x0	PM: 0 = KBC 1 = NAND 2 = TRACE 3 = RSVD

20.1.6.198 PINMUX_AUX_KB_COL6_0

Offset: 3314h | Read/Write: R/W | Reset: 0b0x101000

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	0x0	PM: 0 = KBC 1 = NAND 2 = TRACE 3 = MIO

20.1.6.199 PINMUX_AUX_KB_COL7_0

Offset: 3318h | Read/Write: R/W | Reset: 0b0x101000

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	0x0	PM: 0 = KBC 1 = NAND 2 = TRACE 3 = MIO

20.1.6.200 PINMUX_AUX_CLK_32K_OUT_0

Offset: 331ch | Read/Write: R/W | Reset: 0b0x100000

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	0x0	PM: 0 = BLINK 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.201 PINMUX_AUX_SYS_CLK_REQ_0

Offset: 3320h | Read/Write: R/W | Reset: 0b0x100000

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	0x0	PM: 0 = SYSCLK 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.202 PINMUX_AUX_CORE_PWR_REQ_0

Offset: 3324h | Read/Write: R/W | Reset: 0b0x100000

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	0x0	PM

20.1.6.203 PINMUX_AUX_CPU_PWR_REQ_0

Offset: 3328h | Read/Write: R/W | Reset: 0b0x100000

Bit	Reset	Description
7	DISABLE	LOCK: 0 = DISABLE

Bit	Reset	Description
		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	0x0	PM

20.1.6.204 PINMUX_AUX_PWR_INT_N_0

Offset: 332ch | Read/Write: R/W | Reset: 0b0x100000

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	0x0	PM

20.1.6.205 PINMUX_AUX_CLK_32K_IN_0

Offset: 3330h | Read/Write: R/W | Reset: 0b0x100000

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

Bit	Reset	Description
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM

20.1.6.206 PINMUX_AUX_OWR_0

Offset: 3334h | Read/Write: R/W | Reset: 0b0x101000

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	OWR	PM: ALT1 is CEC as per silicon and PinOutSpec.xls but it is not to be used. 0 = OWR 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.207 PINMUX_AUX_DAP1_FS_0

Offset: 3338h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S0 1 = HDA 2 = GMI 3 = SDMMC2

20.1.6.208 PINMUX_AUX_DAP1_DIN_0

Offset: 333ch | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S0 1 = HDA 2 = GMI 3 = SDMMC2

20.1.6.209 PINMUX_AUX_DAP1_DOUT_0

Offset: 3340h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S0

Bit	Reset	Description
		1 = HDA 2 = GMI 3 = SDMMC2

20.1.6.210 PINMUX_AUX_DAP1_SCLK_0

Offset: 3344h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S0 1 = HDA 2 = GMI 3 = SDMMC2

20.1.6.211 PINMUX_AUX_CLK1_REQ_0

Offset: 3348h | Read/Write: R/W | Reset: 0b0x100000

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	0x0	PM: 0 = DAP 1 = DAP1 2 = RSVD2

Bit	Reset	Description
		3 = RSVD3

20.1.6.212 PINMUX_AUX_CLK1_OUT_0

Offset: 334ch | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = EXTPERIPH1 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.1.6.213 PINMUX_AUX_SPDIF_IN_0

Offset: 3350h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPDIF 1 = DAP2 2 = I2C1 3 = DAPSDMMC2

20.1.6.214 PINMUX_AUX_SPDIF_OUT_0

Offset: 3354h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPDIF 1 = RSVD1 2 = I2C1 3 = DAPSDMMC2

20.1.6.215 PINMUX_AUX_DAP2_FS_0

Offset: 3358h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S1 1 = HDA 2 = RSVD2 3 = GMI

20.1.6.216 PINMUX_AUX_DAP2_DIN_0

Offset: 335ch | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S1 1 = HDA 2 = RSVD2 3 = GMI

20.1.6.217 PINMUX_AUX_DAP2_DOUT_0

Offset: 3360h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S1 1 = HDA 2 = RSVD2 3 = GMI

20.1.6.218 PINMUX_AUX_DAP2_SCLK_0

Offset: 3364h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = I2S1 1 = HDA 2 = RSVD2 3 = GMI

20.1.6.219 PINMUX_AUX_SPI2_MOSI_0

Offset: 3368h | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = SPI6 1 = SPI2 2 = SPI3 3 = GMI

20.1.6.220 PINMUX_AUX_SPI2_MISO_0

Offset: 336ch | Read/Write: R/W | Reset: 0b0x110100

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	0x0	PM: 0 = SPI6 1 = SPI2 2 = SPI3 3 = GMI

20.1.6.221 PINMUX_AUX_SPI2_CS0_N_0

Offset: 3370h | Read/Write: R/W | Reset: 0b0x101000

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	0x0	PM: 0 = SPI6 1 = SPI2 2 = SPI3 3 = GMI

20.1.6.222 PINMUX_AUX_SPI2_SCK_0

Offset: 3374h | Read/Write: R/W | Reset: 0b0x101000

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	0x0	PM: 0 = SPI6 1 = SPI2 2 = SPI3 3 = GMI

20.1.6.223 PINMUX_AUX_SPI1_MOSI_0

Offset: 3378h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPI2 1 = SPI1 2 = SPI2_ALT 3 = GMI

20.1.6.224 PINMUX_AUX_SPI1_SCK_0

Offset: 337ch | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPI2 1 = SPI1 2 = SPI2_ALT 3 = GMI

20.1.6.225 PINMUX_AUX_SPI1_CS0_N_0

Offset: 3380h | Read/Write: R/W | Reset: 0b0x111000

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	0x0	PM: 0 = SPI2 1 = SPI1 2 = SPI2_ALT 3 = GMI

20.1.6.226 PINMUX_AUX_SPI1_MISO_0

Offset: 3384h | Read/Write: R/W | Reset: 0b0x110101

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 = SPI3 1 = SPI1 2 = SPI2 3 = RSVD3

20.1.6.227 PINMUX_AUX_SPI2_CS1_N_0

Offset: 3388h | Read/Write: R/W | Reset: 0b0x111001

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 = SPI3 1 = SPI2 2 = SPI2_ALT 3 = I2C1

20.1.6.228 PINMUX_AUX_SPI2_CS2_N_0

Offset: 338ch | Read/Write: R/W | Reset: 0b0x111001

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 = SPI3 1 = SPI2 2 = SPI2_ALT 3 = I2C1

20.1.6.229 PINMUX_AUX_SDMMC3_CLK_0

Offset: 3390h | Read/Write: R/W | Reset: 0b0x111010

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 = UARTA 1 = PWM2 2 = SDMMC3 3 = SPI3

20.1.6.230 PINMUX_AUX_SDMMC3_CMD_0

Offset: 3394h | Read/Write: R/W | Reset: 0b0x111010

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 = UARTA 1 = PWM3 2 = SDMMC3 3 = SPI2

20.1.6.231 PINMUX_AUX_SDMMC3_DAT0_0

Offset: 3398h | Read/Write: R/W | Reset: 0b0x111010

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 = RSVD0 1 = RSVD1 2 = SDMMC3 3 = SPI3

20.1.6.232 PINMUX_AUX_SDMMC3_DAT1_0

Offset: 339ch | Read/Write: R/W | Reset: 0b0x111010

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 = RSVD0 1 = RSVD1 2 = SDMMC3 3 = SPI3

20.1.6.233 PINMUX_AUX_SDMMC3_DAT2_0

Offset: 33a0h | Read/Write: R/W | Reset: 0b0x111010

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 = RSVD0 1 = PWM1 2 = SDMMC3 3 = SPI3

20.1.6.234 PINMUX_AUX_SDMMC3_DAT3_0

Offset: 33a4h | Read/Write: R/W | Reset: 0b0x111010

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 = RSVD0 1 = PWM0 2 = SDMMC3 3 = SPI3

20.1.6.235 PINMUX_AUX_SDMMC3_DAT4_0

Offset: 33a8h | Read/Write: R/W | Reset: 0b0x111010

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 = PWM1 1 = SPI4 2 = SDMMC3 3 = SPI2

20.1.6.236 PINMUX_AUX_SDMMC3_DAT5_0

Offset: 33ach | Read/Write: R/W | Reset: 0b0x111010

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 = PWM0 1 = SPI4 2 = SDMMC3 3 = SPI2

20.1.6.237 PINMUX_AUX_SDMMC3_DAT6_0

Offset: 33b0h | Read/Write: R/W | Reset: 0b0x111010

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 = SPDIF 1 = SPI4 2 = SDMMC3 3 = SPI2

20.1.6.238 PINMUX_AUX_SDMMC3_DAT7_0

Offset: 33b4h | Read/Write: R/W | Reset: 0b0x111010

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 = SPDIF 1 = SPI4 2 = SDMMC3 3 = SPI2

20.1.6.239 PINMUX_AUX_PEX_L0_PRSNT_N_0

Offset: 33b8h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PCIE 1 = HDA 2 = RSVD2 3 = RSVD3

20.1.6.240 PINMUX_AUX_PEX_L0_RST_N_0

Offset: 33bch | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PCIE 1 = HDA 2 = RSVD2 3 = RSVD3

20.1.6.241 PINMUX_AUX_PEX_L0_CLKREQ_N_0

Offset: 33c0h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PCIE 1 = HDA 2 = RSVD2 3 = RSVD3

20.1.6.242 PINMUX_AUX_PEX_WAKE_N_0

Offset: 33c4h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PCIE 1 = HDA 2 = RSVD2 3 = RSVD3

20.1.6.243 PINMUX_AUX_PEX_L1_PRSNT_N_0

Offset: 33c8h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PCIE 1 = HDA 2 = RSVD2 3 = RSVD3

20.1.6.244 PINMUX_AUX_PEX_L1_RST_N_0

Offset: 33cch | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PCIE 1 = HDA 2 = RSVD2 3 = RSVD3

20.1.6.245 PINMUX_AUX_PEX_L1_CLKREQ_N_0

Offset: 33d0h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PCIE 1 = HDA 2 = RSVD2 3 = RSVD3

20.1.6.246 PINMUX_AUX_PEX_L2_PRSNT_N_0

Offset: 33d4h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PCIE 1 = HDA 2 = RSVD2 3 = RSVD3

20.1.6.247 PINMUX_AUX_PEX_L2_RST_N_0

Offset: 33d8h | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PCIE 1 = HDA 2 = RSVD2 3 = RSVD3

20.1.6.248 PINMUX_AUX_PEX_L2_CLKREQ_N_0

Offset: 33dch | Read/Write: R/W | Reset: 0b0x110000

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	0x0	PM: 0 = PCIE 1 = HDA 2 = RSVD2 3 = RSVD3

20.1.6.249 PINMUX_AUX_HDMI_CEC_0

Offset: 33e0h | Read/Write: R/W | Reset: 0b01100000

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	NORMAL	TRISTATE: 0 = NORMAL 1 = TRISTATE
3:2	NORMAL	PUPD: 0 = NORMAL 1 = PULL_DOWN 2 = PULL_UP 3 = RSVD
1:0	0x0	PM: 0 = CEC 1 = RSVD1 2 = RSVD2 3 = RSVD3

20.2 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 is used for block data transfer 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 to program the parameters for a particular transfer and accesses needed to handle any interrupts.

It has 32 fully programmable channels, which can transfer data concurrently.

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 64KB 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

- The AHB burst size and the FIFO trigger levels (in the modules) should be programmed such that they do not lead to an overflow/underflow of the FIFO.
- SW should program all the registers of channel, ensuring that the channel enable bit is disabled. Channel enable bit should be set last.
- Disable the Global enable bit to halt transfer for some time, and set it when it is all right to resume the transfer. This halts transfers from all channels.
- If channel is disabled while transfer is in progress, the transfer ends after ongoing burst is completed and an interrupt is generated if enabled.
- Busy bit is set as soon as DMA channel is enabled and gets cleared after transfer is completed.
- Interrupts are "write 1 to clear".
- Interrupt is generated when the last data is placed on the AHB bus while writing and once the last data is placed on APB bus while reading from AHB bus.
- The AHB wrap around should be programmed keeping in mind the address that is programmed in the AHB start address and the burst size.

- If, for example, AHB wrap around is 4 words, the AHB start address is 0x4000 0000 and the AHB burst size is NOT 1, then address would change to:

  0x4000 0000 ' 0x4000 0004 ' 0x4000 0008 ' 0x4000 000C ' 0x4000 0000

- For the above condition if the AHB start address is programmed to

  0x0000 0004….

- Then the address would change as

  0x0000 0004 ' 0x0000 0008 ' 0x0000 000C ' 0x0000 0010

  Note that the address did not wrap around! Wrap around is NOT possible in between bursts.

- If this burst size is given to the AHB bus it increments the address 4 times as it does not see "AHB current address" but the initial address that was given to it with a request.
- If the start address has to be specified as 0x0000 0004 with wrap around as needed above, the solution is to give a burst size of 1.

- 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 e.g., base addr, 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 till 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 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 (less than 4 words left to transfer), the DMA completes the remaining transfers with a burst of 1.

20.2.1 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 ch-0.

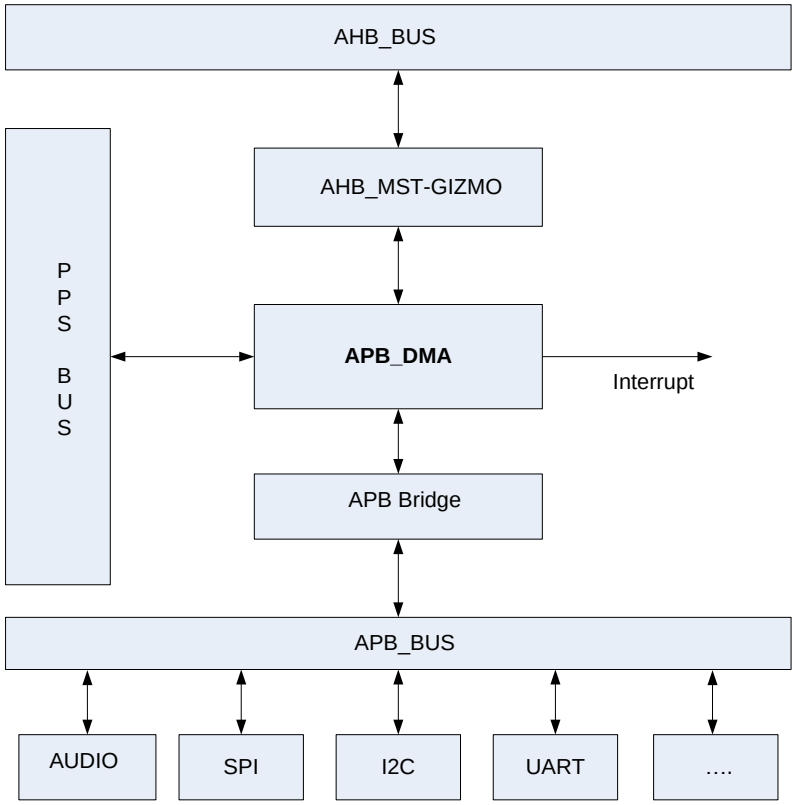
Each channel can have independent burst transfer sizes programmed to one word, four words, or eight words.

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. Processor which 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 enables you to wrap back the LSB nibble to 0000 instead of incrementing to next address if the burst starts from an unaligned address boundary. For example, if address starts at 0x4 with a burst of 4 words, then the address would increment as 0x4, 0x8, 0xC, 0x0. If disabled, it would be 0x4, 0x8, 0xC, 0x10.
- APB bus width is configurable whereas the AHB bus width is fixed to 32 bits. The APB bus can have values 8, 16 and 32. For an APB burst of 8 (burst is always with regard to words) and APB bus size of 16, you have: 8 words transferred to the APB side, or rather 16 "half words" 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 35. APB-DMA Block Diagram



20.2.2 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 Interrupt option, Hold Processor option, DMA transfer direction, Transfer Mode, Trigger/Flow Enable and DMA Count value in CHANNEL_*_CSR Register.
e. Program the Global Enable bit in APB_DMA Command Register.
f. Whenever the Channel ENB bit is Enabled, the DMA starts the Data transfer.
g. Each channel status is observed by polling or interrupt status using APB_DMA Status Register.
h. 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.
3. If channel ENB is disabled independently while a transfer is in progress, the transfer ends after completing any burst sequence that is in progress.
4. Busy bit gets set as soon as a channel is enabled and is cleared after transfer is completed.
5. 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.
6. The AHB 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. If a burst of 4 is requested and at any time the transfer size goes below 4, the DMA completes the remaining transfers with a burst of 1.

20.2.3 APB DMA Registers

20.2.3.1 APBDMA_COMMAND_0

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 (ongoing burst) will be completed and no new transactions will be initiated. Setting this bit again resumes the transfers.

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.

APB-DMA Command Register

Offset: 000h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
31	0x0	GEN: Enables Global APB-DMA 0 = DISABLE 1 = ENABLE

20.2.3.2 APBDMA_STATUS_0

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 HW.

APB-DMA Status Register

Offset: 004h | Read/Write: RO | Reset: 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 1 = BUSY

Bit	Reset	Description
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

Bit	Reset	Description
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 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

20.2.3.3 APBDMA_REQUESTORS_TX_0

The Tx Requestors are values used as triggers or flow controls for APB DMA transfers.

The Tx requestor register is read only. This register is provided only to give the software the ability to see the state of signals that may affect (trigger-start or flow-control-inhibit) APB DMA processes.

Bit[0] in the Tx Requestor channel is a signal generated from the programmable APB DMA Counter. This bit is active whenever the APB DMA Counter value is 0. When Tx request is active, DMA transmits data to peripheral.

APB-DMA Requestor Register (Tx)

Offset: 008h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28	X	SL2B6: SLINK 2B-6 0 = NOT_ACTIVE 1 = ACTIVE
27	X	SL2B5: SLINK 2B-5 0 = NOT_ACTIVE 1 = ACTIVE
26	X	I2C_4: I2C4 0 = NOT_ACTIVE 1 = ACTIVE
25	X	OWR: OWR-I2C 0 = NOT_ACTIVE 1 = ACTIVE
24	X	DVC_I2C: DVC-I2C 0 = NOT_ACTIVE 1 = ACTIVE
23	X	I2C_3: I2C3 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
22	X	I2C_2: I2C2 0 = NOT_ACTIVE 1 = ACTIVE
21	X	I2C_1: I2C1 0 = NOT_ACTIVE 1 = ACTIVE
20	X	UART_E: UARTE 0 = NOT_ACTIVE 1 = ACTIVE
19	X	UART_C: UARTD 0 = NOT_ACTIVE 1 = ACTIVE
18	X	SL2B4: SLINK 2B-4 (SPI4) 0 = NOT_ACTIVE 1 = ACTIVE
17	X	SL2B3: SLINK 2B-3 (SPI3) 0 = NOT_ACTIVE 1 = ACTIVE
16	X	SL2B2: SLINK 2B-2 (SPI2) 0 = NOT_ACTIVE 1 = ACTIVE
15	X	SL2B1: SLINK 2B-1 (SPI1) 0 = NOT_ACTIVE 1 = ACTIVE
10	X	UART_C: UART C 0 = NOT_ACTIVE 1 = ACTIVE
9	X	UART_B: UART B (VFIR) 0 = NOT_ACTIVE 1 = ACTIVE
8	X	UART_A: UART A 0 = NOT_ACTIVE 1 = ACTIVE
5	X	HSI: HSI Tx FIFO 0 = NOT_ACTIVE 1 = ACTIVE
4	X	APBIF_CH3: APBIF_CH3 Tx FIFO (Peripheral initiated DMA request); 1 = Activate DMA transfer; 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
3	X	APBIF_CH2: APBIF_CH2 Tx FIFO (Peripheral initiated DMA request); 1 = Activate DMA transfer; 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
2	X	APBIF_CH1: APBIF_CH1 Tx FIFO (Peripheral initiated DMA request); 1 = Activate DMA transfer; 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
1	X	APBIF_CH0: APBIF_CH0 Tx FIFO (Peripheral initiated DMA request); 1 = Activate DMA transfer; 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
0	X	CNTR_REQ: 1 = Enable counter request. 0 = Disable counter request 0 = NOT_ACTIVE 1 = ACTIVE

20.2.3.4 APBDMA_REQUESTORS_RX_0

The Rx Requestors are values used as triggers or flow controls for APB DMA transfers.

The Rx requestor register is read only. This register is provided only to give the software the ability to see the state of signals that may affect (trigger-start or flow-control-inhibit) APB DMA processes.

Bit[0] in the Rx Requestor channel is a signal generated from the programmable APB DMA Counter. This bit is active whenever the APB DMA Counter value is 0. When Rx request is active, DMA receives data from peripheral.

APB-DMA Requestor Register (RX)

Offset: 00ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28	X	SL2B6: SLINK 2B-6 0 = NOT_ACTIVE 1 = ACTIVE
27	X	SL2B5: SLINK 2B-5 0 = NOT_ACTIVE 1 = ACTIVE
26	X	I2C_4: I2C4 0 = NOT_ACTIVE 1 = ACTIVE
25	X	OWR: OWR-I2C 0 = NOT_ACTIVE 1 = ACTIVE
24	X	DVC_I2C: DVC-I2C 0 = NOT_ACTIVE 1 = ACTIVE
23	X	I2C_3: I2C3 0 = NOT_ACTIVE 1 = ACTIVE
22	X	I2C_2: I2C2 0 = NOT_ACTIVE 1 = ACTIVE
21	X	I2C_1: I2C1 0 = NOT_ACTIVE 1 = ACTIVE
20	X	UART_E: UARTE 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
19	X	UART_C: UARTD 0 = NOT_ACTIVE 1 = ACTIVE
18	X	SL2B4: SLINK 2B-4 (SPI4) 0 = NOT_ACTIVE 1 = ACTIVE
17	X	SL2B3: SLINK 2B-3 (SPI3) 0 = NOT_ACTIVE 1 = ACTIVE
16	X	SL2B2: SLINK 2B-2 (SPI2) 0 = NOT_ACTIVE 1 = ACTIVE
15	X	SL2B1: SLINK 2B-1 (SPI1) 0 = NOT_ACTIVE 1 = ACTIVE
11	X	SPI: SPI/DTV 0 = NOT_ACTIVE 1 = ACTIVE
10	X	UART_C: UART C 0 = NOT_ACTIVE 1 = ACTIVE
9	X	UART_B: UART B (VFIR) 0 = NOT_ACTIVE 1 = ACTIVE
8	X	UART_A: UART A 0 = NOT_ACTIVE 1 = ACTIVE
5	X	HSI: HSI Rx FIFO 0 = NOT_ACTIVE 1 = ACTIVE
4	X	APBIF_CH3: APBIF_CH3 Rx FIFO (Peripheral initiated DMA request) 1 = Activate DMA transfer; 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
3	X	APBIF_CH2: APBIF_CH2 Rx FIFO (Peripheral initiated DMA request) 1 = Activate DMA transfer; 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
2	X	APBIF_CH1: APBIF_CH1 Rx FIFO (Peripheral initiated DMA request) 1 = Activate DMA transfer; 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE
1	X	APBIF_CH0: APBIF_CH0 Rx FIFO (Peripheral initiated DMA request) 1 = Activate DMA transfer; 0 = NOP 0 = NOT_ACTIVE 1 = ACTIVE

Bit	Reset	Description
0	X	CNTR_REQ: indicates Enabled counter request or not 0 = NOT_ACTIVE 1 = ACTIVE

20.2.3.5 APBDMA_CNTRL_REG_0

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.

APB-DMA Counter Register

Offset: 010h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	COUNT_VALUE: DMA COUNT Value.

20.2.3.6 APBDMA_IRQ_STA_CPU_0

Gathers all the after-masking CPU directed IRQ status bits.

APB-DMA CPU IRQ Status Register

Offset: 014h | Read/Write: RO | Reset: 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

Bit	Reset	Description
25	X	CH25: Gathers all the after-masking CPU directed IRQ status bits from channel25 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

Bit	Reset	Description
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 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

20.2.3.7 APBDMA_IRQ_STA_COP_0

Gathers all the after-masking COP directed IRQ status bits.

APB-DMA COP IRQ Status Register

Offset: 018h | Read/Write: RO | Reset: 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

Bit	Reset	Description
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
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

Bit	Reset	Description
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
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

20.2.3.8 APBDMA_IRQ_MASK_0

Allows the IRQ to propagate when enabled, this is the ANDed result of set and clear.

APB-DMA IRQ Mask Register

Offset: 01ch | Read/Write: RO | Reset: 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
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

Bit	Reset	Description
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
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

Bit	Reset	Description
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'

20.2.3.9 APBDMA_IRQ_MASK_SET_0

APB-DMA IRQ Mask Set Register

Offset: 020h | Read/Write: WO | Reset: 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 0 = DISABLE 1 = ENABLE
21	0x0	CH21: Sets the Mask Register 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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

Bit	Reset	Description
5	0x0	CH5: Sets the Mask Register 0 = DISABLE 1 = ENABLE
4	0x0	CH4: Sets the Mask Register 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

20.2.3.10 APBDMA_IRQ_MASK_CLR_0

APB-DMA IRQ Mask Clear Register

Offset: 024h | Read/Write: WO | Reset: 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

Bit	Reset	Description
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 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

Bit	Reset	Description
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 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

20.2.3.11 APBDMA_TRIG_REG_0

APB-DMA Trigger Register

Offset: 028h | Read/Write: RO | Reset: 0bxxxxxxxx

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

Bit	Reset	Description
5	X	XRQ_A: XRQ.A (GPIOA) (Hardware initiated DMA request) 0 = NOT_ACTIVE 1 = ACTIVE
4	X	SMP_27: Semaphore requests SW initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
3	X	SMP_26: Semaphore requests SW initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
2	X	SMP_25: Semaphore requests SW initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE
1	X	SMP_24: Semaphore requests SW initiated DMA request 0 = NOT_ACTIVE 1 = ACTIVE

20.2.3.12 APBDMA_CHANNEL_TRIG_REG_0

APB-DMA Channel Trigger registers

Offset: 02ch l Read/Write: RO l Reset: 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 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

Bit	Reset	Description
		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 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

Bit	Reset	Description
		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

20.2.3.13 APBDMA_DMA_STATUS_0

APB-DMA Interrupt Status

Offset: 030h l Read/Write: RO l Reset: 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 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

Bit	Reset	Description
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 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

Bit	Reset	Description
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

20.2.3.14 APBDMA_CHANNEL_EN_REG_0

APB-DMA Channel Counter Enable registers

Offset: 034h l Read/Write: R/W l Reset: 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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
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

20.2.4 APB DMA Channel Registers

20.2.4.1 APBDMACHAN_CHANNEL_0_CSR_0

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 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.

APB-DMA-0 Control Register

Offset: 000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.2 APBDMACHAN_CHANNEL_0_STA_0

APB-DMA-0 Status Register

Offset: 004h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status 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_STS: 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_STS 1 = PONG_INTR_STS

Bit	R/W	Reset	Description
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.3 APBDMACHAN_CHANNEL_0_DMA_BYTE_STA_0

APB-DMA-0 DMA Byte Status Register

Offset: 008h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.4 APBDMACHAN_CHANNEL_0_AHB_PTR_0

APB-DMA-0 AHB Starting Address Pointer Register

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.5 APBDMACHAN_CHANNEL_0_AHB_SEQ_0

APB-DMA-0 AHB Address Sequencer Register

Offset: 014h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP
30:28	0x2	AHB_BUS_WIDTH: AHB Bus Width 0 = 8 bit Bus (RSVD) 1 = 16 bit Bus (RSVD) 2 = 32 bit Bus (Def) 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 (4x32bits) else = 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 (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
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

20.2.4.6 APBDMACHAN_CHANNEL_0_APB_PTR_0

APB-DMA-0 APB Starting Address Pointer Register

Offset: 018h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.7 APBDMACHAN_CHANNEL_0_APB_SEQ_0

APB-DMA-0 APB Address Sequencer Assignments

Offset: 01ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]}. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.8 APBDMACHAN_CHANNEL_1_CSR_0

APB-DMA-1 Control

Offset: 020h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.9 APBDMACHAN_CHANNEL_1_STA_0

APB-DMA-1 Status Register

Offset: 024h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status 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_STS: 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_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.10 APBDMACHAN_CHANNEL_1_DMA_BYTE_STA_0

APB-DMA-1 DMA Byte Status Register

Offset: 028h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.11 APBDMACHAN_CHANNEL_1_AHB_PTR_0

APB-DMA-1 AHB Starting Address Pointer Register

Offset: 030h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.12 APBDMACHAN_CHANNEL_1_AHB_SEQ_0

APB-DMA-1 AHB Address Sequencer Register

Offset: 034h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
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

20.2.4.13 APBDMACHAN_CHANNEL_1_APB_PTR_0

APB-DMA-1 APB Starting Address Pointer Register

Offset: 038h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.14 APBDMACHAN_CHANNEL_1_APB_SEQ_0

APB-DMA-1 APB Address Sequencer Assignments

Offset: 03ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.15 APBDMACHAN_CHANNEL_2_CSR_0

APB-DMA-2 Control

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.16 APBDMACHAN_CHANNEL_2_STA_0

APB-DMA-2 Status Register

Offset: 044h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status 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_STS: 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_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.17 APBDMACHAN_CHANNEL_2_DMA_BYTE_STA_0

APB-DMA-2 DMA Byte Status Register

Offset: 048h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.18 APBDMACHAN_CHANNEL_2_AHB_PTR_0

APB-DMA-2 AHB Starting Address Pointer Register

Offset: 050h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.19 APBDMACHAN_CHANNEL_2_AHB_SEQ_0

APB-DMA-2 AHB Address Sequencer Register

Offset: 054h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
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

20.2.4.20 APBDMACHAN_CHANNEL_2_APB_PTR_0

APB-DMA-2 APB Starting Address Pointer Register

Offset: 058h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.21 APBDMACHAN_CHANNEL_2_APB_SEQ_0

APB-DMA-2 APB Address Sequencer Assignments

Offset: 05ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.22 APBDMACHAN_CHANNEL_3_CSR_0

APB-DMA-3 Control

Offset: 060h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.23 APBDMACHAN_CHANNEL_3_STA_0

APB-DMA-3 Status Register

Offset: 064h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status active 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_STS: 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_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.24 APBDMACHAN_CHANNEL_3_DMA_BYTE_STA_0

APB-DMA-3 DMA Byte Status Register

Offset: 068h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.25 APBDMACHAN_CHANNEL_3_AHB_PTR_0

APB-DMA-3 AHB Starting Address Pointer Register

Offset: 070h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.26 APBDMACHAN_CHANNEL_3_AHB_SEQ_0

APB-DMA-3 AHB Address Sequencer Register

Offset: 074h | Read/Write: R/W | Reset: 0b00100110xxxxx000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS

Bit	Reset	Description
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

20.2.4.27 APBDMACHAN_CHANNEL_3_APB_PTR_0

APB-DMA-3 APB Starting Address Pointer Register

Offset: 078h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.28 APBDMACHAN_CHANNEL_3_APB_SEQ_0

APB-DMA-3 APB Address Sequencer Assignments

Offset: 07ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] } 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.29 APBDMACHAN_CHANNEL_4_CSR_0

APB-DMA-4 Control

Offset: 080h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.30 APBDMACHAN_CHANNEL_4_STA_0

APB-DMA-4 Status Register

Offset: 084h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status 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_STS: 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_STS 1 = PONG_INTR_STS

Bit	R/W	Reset	Description
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.31 APBDMACHAN_CHANNEL_4_DMA_BYTE_STA_0

APB-DMA-4 DMA Byte Status Register

Offset: 088h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.32 APBDMACHAN_CHANNEL_4_AHB_PTR_0

APB-DMA-4 AHB Starting Address Pointer Register

Offset: 090h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.33 APBDMACHAN_CHANNEL_4_AHB_SEQ_0

APB-DMA-4 AHB Address Sequencer Register

Offset: 094h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS

Bit	Reset	Description
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 (def) (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

20.2.4.34 APBDMACHAN_CHANNEL_4_APB_PTR_0

APB-DMA-4 APB Starting Address Pointer Register

Offset: 098h | Read/Write: R/W | Reset: 0b0000000000000000000

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

20.2.4.35 APBDMACHAN_CHANNEL_4_APB_SEQ_0

APB-DMA-4 APB Address Sequencer Assignments

Offset: 09ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.36 APBDMACHAN_CHANNEL_5_CSR_0

APB-DMA-5 Control

Offset: 0a0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.37 APBDMACHAN_CHANNEL_5_STA_0

APB-DMA-5 Status Register

Offset: 0a4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate whether DMA Channel Status 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_STS: 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_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.38 APBDMACHAN_CHANNEL_5_DMA_BYTE_STA_0

APB-DMA-5 DMA Byte Status Register

Offset: 0a8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.39 APBDMACHAN_CHANNEL_5_AHB_PTR_0

APB-DMA-5 AHB Starting Address Pointer Register

Offset: 0b0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.40 APBDMACHAN_CHANNEL_5_AHB_SEQ_0

APB-DMA-5 AHB Address Sequencer Register

Offset: 0b4h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
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

20.2.4.41 APBDMACHAN_CHANNEL_5_APB_PTR_0

APB-DMA-5 APB Starting Address Pointer Register

Offset: 0b8h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.42 APBDMACHAN_CHANNEL_5_APB_SEQ_0

APB-DMA-5 APB Address Sequencer Assignments

Offset: 0bch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.43 APBDMACHAN_CHANNEL_6_CSR_0

APB-DMA-6 Control

Offset: 0c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.44 APBDMACHAN_CHANNEL_6_STA_0

APB-DMA-6 Status Register

Offset: 0c4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate whether DMA Channel Status 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_STS: 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_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.45 APBDMACHAN_CHANNEL_6_DMA_BYTE_STA_0

APB-DMA-6 DMA Byte Status Register

Offset: 0c8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.46 APBDMACHAN_CHANNEL_6_AHB_PTR_0

APB-DMA-6 AHB Starting Address Pointer Register

Offset: 0d0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.47 APBDMACHAN_CHANNEL_6_AHB_SEQ_0

APB-DMA-6 AHB Address Sequencer Register

Offset: 0d4h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
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

20.2.4.48 APBDMACHAN_CHANNEL_6_APB_PTR_0

APB-DMA-6 APB Starting Address Pointer Register

Offset: 0d8h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.49 APBDMACHAN_CHANNEL_6_APB_SEQ_0

APB-DMA-6 APB Address Sequencer Assignments

Offset: 0dch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.50 APBDMACHAN_CHANNEL_7_CSR_0

APB-DMA-7 Control

Offset: 0e0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.51 APBDMACHAN_CHANNEL_7_STA_0

APB-DMA-7 Status Register

Offset: 0e4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate whether DMA Channel Status 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_STS: 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_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.52 APBDMACHAN_CHANNEL_7_DMA_BYTE_STA_0

APB-DMA-7 DMA Byte Status Register

Offset: 0e8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.53 APBDMACHAN_CHANNEL_7_AHB_PTR_0

APB-DMA-7 AHB Starting Address Pointer Register

Offset: 0f0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.54 APBDMACHAN_CHANNEL_7_AHB_SEQ_0

APB-DMA-7 AHB Address Sequencer Register

Offset: 0f4h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
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

20.2.4.55 APBDMACHAN_CHANNEL_7_APB_PTR_0

APB-DMA-7 APB Starting Address Pointer Register

Offset: 0f8h | Read/Write: R/W | Reset: 0b00000000000000000

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

20.2.4.56 APBDMACHAN_CHANNEL_7_APB_SEQ_0

APB-DMA-7 APB Address Sequencer Assignments

Offset: 0fch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]}. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.57 APBDMACHAN_CHANNEL_8_CSR_0

APB-DMA-8 Control

Offset: 100h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.58 APBDMACHAN_CHANNEL_8_STA_0

APB-DMA-8 Status Register

Offset: 104h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate 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_STS: 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_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.59 APBDMACHAN_CHANNEL_8_DMA_BYTE_STA_0

APB-DMA-8 DMA Byte Status Register

Offset: 108h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.60 APBDMACHAN_CHANNEL_8_AHB_PTR_0

APB-DMA-8 AHB Starting Address Pointer Register

Offset: 110h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.61 APBDMACHAN_CHANNEL_8_AHB_SEQ_0

APB-DMA-8 AHB Address Sequencer Register

Offset: 114h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
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

20.2.4.62 APBDMACHAN_CHANNEL_8_APB_PTR_0

APB-DMA-8 APB Starting Address Pointer Register

Offset: 118h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.63 APBDMACHAN_CHANNEL_8_APB_SEQ_0

APB-DMA-8 APB Address Sequencer Assignments

Offset: 11ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] }. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.64 APBDMACHAN_CHANNEL_9_CSR_0

APB-DMA-9 Control

Offset: 120h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.65 APBDMACHAN_CHANNEL_9_STA_0

APB-DMA-9 Status Register

Offset: 124h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate 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_STS: 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_STS 1 = PONG_INTR_STS

Bit	R/W	Reset	Description
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.66 APBDMACHAN_CHANNEL_9_DMA_BYTE_STA_0

APB-DMA-9 DMA Byte Status Register

Offset: 128h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.67 APBDMACHAN_CHANNEL_9_AHB_PTR_0

APB-DMA-9 AHB Starting Address Pointer Register

Offset: 130h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.68 APBDMACHAN_CHANNEL_9_AHB_SEQ_0

APB-DMA-9 AHB Address Sequencer Register

Offset: 134h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS

Bit	Reset	Description
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 (def) (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

20.2.4.69 APBDMACHAN_CHANNEL_9_APB_PTR_0

APB-DMA-9 APB Starting Address Pointer Register

Offset: 138h | Read/Write: R/W | Reset: 0b0000000000000000000

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

20.2.4.70 APBDMACHAN_CHANNEL_9_APB_SEQ_0

APB-DMA-9 APB Address Sequencer Assignments

Offset: 13ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24] } 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.71 APBDMACHAN_CHANNEL_10_CSR_0

APB-DMA-10 Control

Offset: 140h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold this Processor until DMA Block Transfer Completes 1 = Enable Hold (wait) 0 = Disable 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.72 APBDMACHAN_CHANNEL_10_STA_0

APB-DMA-10 Status Register

Offset: 144h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate 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_STS: 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_STS 1 = PONG_INTR_STS

Bit	R/W	Reset	Description
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.73 APBDMACHAN_CHANNEL_10_DMA_BYTE_STA_0

APB-DMA-10 DMA Byte Status Register

Offset: 148h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.74 APBDMACHAN_CHANNEL_10_AHB_PTR_0

APB-DMA-10 AHB Starting Address Pointer Register

Offset: 150h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.75 APBDMACHAN_CHANNEL_10_AHB_SEQ_0

APB-DMA-10 AHB Address Sequencer Register

Offset: 154h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS

Bit	Reset	Description
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 (def) (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

20.2.4.76 APBDMACHAN_CHANNEL_10_APB_PTR_0

APB-DMA-10 APB Starting Address Pointer Register

Offset: 158h | Read/Write: R/W | Reset: 0b0000000000000000000

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

20.2.4.77 APBDMACHAN_CHANNEL_10_APB_SEQ_0

APB-DMA-10 APB Address Sequencer Assignments

Offset: 15ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.78 APBDMACHAN_CHANNEL_11_CSR_0

APB-DMA-11 Control

Offset: 160h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.79 APBDMACHAN_CHANNEL_11_STA_0

APB-DMA-11 Status Register

Offset: 164h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate or waiting 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_STS: 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_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.80 APBDMACHAN_CHANNEL_11_DMA_BYTE_STA_0

APB-DMA-11 DMA Byte Status Register

Offset: 168h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.81 APBDMACHAN_CHANNEL_11_AHB_PTR_0

APB-DMA-11 AHB Starting Address Pointer Register

Offset: 170h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.82 APBDMACHAN_CHANNEL_11_AHB_SEQ_0

APB-DMA-11 AHB Address Sequencer Register

Offset: 174h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits) else = 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 (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
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

20.2.4.83 APBDMACHAN_CHANNEL_11_APB_PTR_0

APB-DMA-11 APB Starting Address Pointer Register

Offset: 178h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.84 APBDMACHAN_CHANNEL_11_APB_SEQ_0

APB-DMA-11 APB Address Sequencer Assignments

Offset: 17ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.85 APBDMACHAN_CHANNEL_12_CSR_0

APB-DMA-12 Control

Offset: 180h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.86 APBDMACHAN_CHANNEL_12_STA_0

APB-DMA-12 Status Register

Offset: 184h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate 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_STS: 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_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32-bit word cycles Flags set /cleared by HW

20.2.4.87 APBDMACHAN_CHANNEL_12_DMA_BYTE_STA_0

APB-DMA-12 DMA Byte Status Register

Offset: 188h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.88 APBDMACHAN_CHANNEL_12_AHB_PTR_0

APB-DMA-12 AHB Starting Address Pointer Register

Offset: 190h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.89 APBDMACHAN_CHANNEL_12_AHB_SEQ_0

APB-DMA-12 AHB Address Sequencer Register

Offset: 194h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
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

20.2.4.90 APBDMACHAN_CHANNEL_12_APB_PTR_0

APB-DMA-12 APB Starting Address Pointer Register

Offset: 198h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.91 APBDMACHAN_CHANNEL_12_APB_SEQ_0

APB-DMA-12 APB Address Sequencer Assignments

Offset: 19ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.92 APBDMACHAN_CHANNEL_13_CSR_0

APB-DMA-13 Control

Offset: 1a0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32-bit word cycles

20.2.4.93 APBDMACHAN_CHANNEL_13_STA_0

APB-DMA-13 Status Register

Offset: 1a4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate 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_STS: 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_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32-bit word cycles Flags set /cleared by HW

20.2.4.94 APBDMACHAN_CHANNEL_13_DMA_BYTE_STA_0

APB-DMA-13 DMA Byte Status Register

Offset: 1a8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.95 APBDMACHAN_CHANNEL_13_AHB_PTR_0

APB-DMA-13 AHB Starting Address Pointer Register

Offset: 1b0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.96 APBDMACHAN_CHANNEL_13_AHB_SEQ_0

APB-DMA-13 AHB Address Sequencer Register

Offset: 1b4h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
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

20.2.4.97 APBDMACHAN_CHANNEL_13_APB_PTR_0

APB-DMA-13 APB Starting Address Pointer Register

Offset: 1b8h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.98 APBDMACHAN_CHANNEL_13_APB_SEQ_0

APB-DMA-13 APB Address Sequencer Assignments

Offset: 1bch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0= DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.99 APBDMACHAN_CHANNEL_14_CSR_0

APB-DMA-14 Control

Offset: 1c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.100 APBDMACHAN_CHANNEL_14_STA_0

APB-DMA-14 Status Register

Offset: 1c4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate 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_STS: 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_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.101 APBDMACHAN_CHANNEL_14_DMA_BYTE_STA_0

APB-DMA-14 DMA Byte Status Register

Offset: 1c8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.102 APBDMACHAN_CHANNEL_14_AHB_PTR_0

APB-DMA-14 AHB Starting Address Pointer Register

Offset: 1d0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.103 APBDMACHAN_CHANNEL_14_AHB_SEQ_0

APB-DMA-14 AHB Address Sequencer Register

Offset: 1d4h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
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

20.2.4.104 APBDMACHAN_CHANNEL_14_APB_PTR_0

APB-DMA-14 APB Starting Address Pointer Register

Offset: 1d8h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.105 APBDMACHAN_CHANNEL_14_APB_SEQ_0

APB-DMA-14 APB Address Sequencer Assignments

Offset: 1dch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to APB gets swapped as [31:0] --> {[7:0], [15:8], [23:16], [31:24]} 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x1	APB_ADDR_WRAP: APB Address Wrap-around Window 0 = No Wrap 1 = Wrap on 1 Word Window (def) 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 Window (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

20.2.4.106 APBDMACHAN_CHANNEL_15_CSR_0

APB-DMA-15 Control

Offset: 1e0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enable DMA channel transfer 0 = DISABLE 1 = ENABLE
30	0x0	IE_EOC: Interrupt when DMA Block Transfer Completes 0 = DISABLE 1 = ENABLE
29	0x0	HOLD: Hold 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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK

Bit	Reset	Description
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
21	0x0	FLOW: Flow Control Enable (Synchronize Burst Transfers) 1 = Link to DRQ source 0 = Independent of DRQ request 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
20:16	0x0	REQ_SEL: 0 = CNTR_REQ 1 = APBIF_CH0 2 = APBIF_CH1 3 = APBIF_CH2 4 = APBIF_CH3 5 = HSI 6 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.107 APBDMACHAN_CHANNEL_15_STA_0

APB-DMA-15 Status Register

Offset: 1e4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate 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_STS: 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_STS 1 = PONG_INTR_STS
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.108 APBDMACHAN_CHANNEL_15_DMA_BYTE_STA_0

APB-DMA-15 DMA Byte Status Register

Offset: 1e8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.109 APBDMACHAN_CHANNEL_15_AHB_PTR_0

APB-DMA-15 AHB Starting Address Pointer Register

Offset: 1f0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.110 APBDMACHAN_CHANNEL_15_AHB_SEQ_0

APB-DMA-15 AHB Address Sequencer Register

Offset: 1f4h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (reload each time) 0 = RELOAD_FOR_1X_BLOCKS 1 = RELOAD_FOR_2X_BLOCKS

Bit	Reset	Description
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

20.2.4.111 APBDMACHAN_CHANNEL_15_APB_PTR_0

APB-DMA-15 APB Starting Address Pointer Register

Offset: 1f8h | Read/Write: R/W | Reset: 0b00000000000000000

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

20.2.4.112 APBDMACHAN_CHANNEL_15_APB_SEQ_0

APB-DMA-15 APB Address Sequencer Assignments

Offset: 1fch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to 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 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

20.2.4.113 APBDMACHAN_CHANNEL_16_CSR_0

APB-DMA-16 Control register

Offset: 200h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
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 = NA6 7 = NA7 8 = UART_A

Bit	Reset	Description
		9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.114 APBDMACHAN_CHANNEL_16_STA_0

APB-DMA-16 Status Register

Offset: 204h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status 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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.115 APBDMACHAN_CHANNEL_16_DMA_BYTE_STA_0

APB-DMA-16 DMA Byte Status Register

Offset: 208h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.116 APBDMACHAN_CHANNEL_16_AHB_PTR_0

APB-DMA-16 AHB Starting Address Pointer Register

Offset: 210h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.117 APBDMACHAN_CHANNEL_16_AHB_SEQ_0

APB-DMA-16 AHB Address Sequencer Register

Offset: 214h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (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

20.2.4.118 APBDMACHAN_CHANNEL_16_APB_PTR_0

APB-DMA-16 APB Starting Address Pointer Register

Offset: 218h | Read/Write: R/W | Reset: 0b00000000000000000000

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

20.2.4.119 APBDMACHAN_CHANNEL_16_APB_SEQ_0

APB-DMA-16 APB Address Sequencer Assignments

Offset: 21ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus. 1 = 16 bit Bus. 2 = 32-bit Bus (Def). 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	APB_DATA_SWAP: When enabled the data going to 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 (def) 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 Window (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

20.2.4.120 APBDMACHAN_CHANNEL_17_CSR_0

APB-DMA-17 Control

Offset: 220h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer

Bit	Reset	Description
		0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
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 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31

Bit	Reset	Description
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.121 APBDMACHAN_CHANNEL_17_STA_0

APB-DMA-17 Status Register

Offset: 224h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status 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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.122 APBDMACHAN_CHANNEL_17_DMA_BYTE_STA_0

APB-DMA-17 DMA Byte Status Register

Offset: 228h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.123 APBDMACHAN_CHANNEL_17_AHB_PTR_0

APB-DMA-17 AHB Starting Address Pointer Register

Offset: 230h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.124 APBDMACHAN_CHANNEL_17_AHB_SEQ_0

APB-DMA-17 AHB Address Sequencer Register

Offset: 234h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (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

20.2.4.125 APBDMACHAN_CHANNEL_17_APB_PTR_0

APB-DMA-1 APB Starting Address Pointer Register

Offset: 238h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.126 APBDMACHAN_CHANNEL_17_APB_SEQ_0

APB-DMA-17 APB Address Sequencer Assignments

Offset: 23ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: when enabled the data going to 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 (def) 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 Window (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

20.2.4.127 APBDMACHAN_CHANNEL_18_CSR_0

APB-DMA-18 Control

Offset: 240h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25

Bit	Reset	Description
		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
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 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.128 APBDMACHAN_CHANNEL_18_STA_0

APB-DMA-18 Status Register

Offset: 244h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status 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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.129 APBDMACHAN_CHANNEL_18_DMA_BYTE_STA_0

APB-DMA-18 DMA Byte Status Register

Offset: 248h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.130 APBDMACHAN_CHANNEL_18_AHB_PTR_0

APB-DMA-18 AHB Starting Address Pointer Register

Offset: 250h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.131 APBDMACHAN_CHANNEL_18_AHB_SEQ_0

APB-DMA-18 AHB Address Sequencer Register

Offset: 254h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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	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 (4x32bits)else = 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 (def) (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

20.2.4.132 APBDMACHAN_CHANNEL_18_APB_PTR_0

APB-DMA-18 APB Starting Address Pointer Register

Offset: 258h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.133 APBDMACHAN_CHANNEL_18_APB_SEQ_0

APB-DMA-18 APB Address Sequencer Assignments

Offset: 25ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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

Bit	Reset	Description
27	0x0	APB_DATA_SWAP: when enabled the data going to 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 (def) 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 Window (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

20.2.4.134 APBDMACHAN_CHANNEL_19_CSR_0

APB-DMA-19 Control

Offset: 260h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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

Bit	Reset	Description
		18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
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 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.135 APBDMACHAN_CHANNEL_19_STA_0

APB-DMA-19 Status Register

Offset: 264h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status active 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

Bit	R/W	Reset	Description
			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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.136 APBDMACHAN_CHANNEL_19_DMA_BYTE_STA_0

APB-DMA-19 DMA Byte Status Register

Offset: 268h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.137 APBDMACHAN_CHANNEL_19_AHB_PTR_0

APB-DMA-19 AHB Starting Address Pointer Register

Offset: 270h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.138 APBDMACHAN_CHANNEL_19_AHB_SEQ_0

APB-DMA-19 AHB Address Sequencer Register

Offset: 274h | Read/Write: R/W | Reset: 0b00100110xxxxx000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS

Bit	Reset	Description
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

20.2.4.139 APBDMACHAN_CHANNEL_19_APB_PTR_0

APB-DMA-19 APB Starting Address Pointer Register

Offset: 278h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.140 APBDMACHAN_CHANNEL_19_APB_SEQ_0

APB-DMA-19 APB Address Sequencer Assignments

Offset: 27ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: when enabled the data going to 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 (def) 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 Window (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

20.2.4.141 APBDMACHAN_CHANNEL_20_CSR_0

APB-DMA-20 Control

Offset: 280h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
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 = NA6

Bit	Reset	Description
		7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.142 APBDMACHAN_CHANNEL_20_STA_0

APB-DMA-20 Status Register

Offset: 284h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicates whether DMA Channel Status 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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.143 APBDMACHAN_CHANNEL_20_DMA_BYTE_STA_0

APB-DMA-20 DMA Byte Status Register

Offset: 288h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.144 APBDMACHAN_CHANNEL_20_AHB_PTR_0

APB-DMA-20 AHB Starting Address Pointer Register

Offset: 290h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.145 APBDMACHAN_CHANNEL_20_AHB_SEQ_0

APB-DMA-20 AHB Address Sequencer Register

Offset: 294h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (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

Bit	Reset	Description
		4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

20.2.4.146 APBDMACHAN_CHANNEL_20_APB_PTR_0

APB-DMA-20 APB Starting Address Pointer Register

Offset: 298h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.147 APBDMACHAN_CHANNEL_20_APB_SEQ_0

APB-DMA-20 APB Address Sequencer Assignments

Offset: 29ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: when enabled the data going to 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 (def) 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 Window (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

20.2.4.148 APBDMACHAN_CHANNEL_21_CSR_0

APB-DMA-21 Control

Offset: 2a0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enables DMA channel transfer 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
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 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3

Bit	Reset	Description
		18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.149 APBDMACHAN_CHANNEL_21_STA_0

APB-DMA-21 Status Register

Offset: 2a4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate whether DMA Channel Status 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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.150 APBDMACHAN_CHANNEL_21_DMA_BYTE_STA_0

APB-DMA-21 DMA Byte Status Register

Offset: 2a8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.151 APBDMACHAN_CHANNEL_21_AHB_PTR_0

APB-DMA-21 AHB Starting Address Pointer Register

Offset: 2b0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.152 APBDMACHAN_CHANNEL_21_AHB_SEQ_0

APB-DMA-21 AHB Address Sequencer Register

Offset: 2b4h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (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

20.2.4.153 APBDMACHAN_CHANNEL_21_APB_PTR_0

APB-DMA-21 APB Starting Address Pointer Register

Offset: 2b8h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.154 APBDMACHAN_CHANNEL_21_APB_SEQ_0

APB-DMA-21 APB Address Sequencer Assignments

Offset: 2bch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: when enabled the data going to 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 (def) 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 Window (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

20.2.4.155 APBDMACHAN_CHANNEL_22_CSR_0

APB-DMA-22 Control

Offset: 2c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25

Bit	Reset	Description
		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
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 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.156 APBDMACHAN_CHANNEL_22_STA_0

APB-DMA-22 Status Register

Offset: 2c4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate whether DMA Channel Status 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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.157 APBDMACHAN_CHANNEL_22_DMA_BYTE_STA_0

APB-DMA-22 DMA Byte Status Register

Offset: 2c8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.158 APBDMACHAN_CHANNEL_22_AHB_PTR_0

APB-DMA-22 AHB Starting Address Pointer Register

Offset: 2d0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.159 APBDMACHAN_CHANNEL_22_AHB_SEQ_0

APB-DMA-22 AHB Address Sequencer Register

Offset: 2d4h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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	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 (4x32bits)else = 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 (def) (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

20.2.4.160 APBDMACHAN_CHANNEL_22_APB_PTR_0

APB-DMA-22 APB Starting Address Pointer Register

Offset: 2d8h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.161 APBDMACHAN_CHANNEL_22_APB_SEQ_0

APB-DMA-22 APB Address Sequencer Assignments

Offset: 2dch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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

Bit	Reset	Description
27	0x0	APB_DATA_SWAP: when enabled the data going to 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 (def) 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 Window (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

20.2.4.162 APBDMACHAN_CHANNEL_23_CSR_0

APB-DMA-23 Control

Offset: 2e0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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

Bit	Reset	Description
		18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
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 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.163 APBDMACHAN_CHANNEL_23_STA_0

APB-DMA-23 Status Register

Offset: 2e4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate whether DMA Channel Status 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

Bit	R/W	Reset	Description
			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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.164 APBDMACHAN_CHANNEL_23_DMA_BYTE_STA_0

APB-DMA-23 DMA Byte Status Register

Offset: 2e8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.165 APBDMACHAN_CHANNEL_23_AHB_PTR_0

APB-DMA-23 AHB Starting Address Pointer Register

Offset: 2f0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.166 APBDMACHAN_CHANNEL_23_AHB_SEQ_0

APB-DMA-23 AHB Address Sequencer Register

Offset: 2f4h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS

Bit	Reset	Description
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 (def) (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

20.2.4.167 APBDMACHAN_CHANNEL_23_APB_PTR_0

APB-DMA-23 APB Starting Address Pointer Register

Offset: 2f8h | Read/Write: R/W | Reset: 0b0000000000000000000

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

20.2.4.168 APBDMACHAN_CHANNEL_23_APB_SEQ_0

APB-DMA-23 APB Address Sequencer Assignments

Offset: 2fch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: when enabled the data going to 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 (def) 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 Window (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

20.2.4.169 APBDMACHAN_CHANNEL_24_CSR_0

APB-DMA-24 Control

Offset: 300h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
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 = NA6

Bit	Reset	Description
		7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.170 APBDMACHAN_CHANNEL_24_STA_0

APB-DMA-24 Status Register

Offset: 304h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate 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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.171 APBDMACHAN_CHANNEL_24_DMA_BYTE_STA_0

APB-DMA-24 DMA Byte Status Register

Offset: 308h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.172 APBDMACHAN_CHANNEL_24_AHB_PTR_0

APB-DMA-24 AHB Starting Address Pointer Register

Offset: 310h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.173 APBDMACHAN_CHANNEL_24_AHB_SEQ_0

APB-DMA-24 AHB Address Sequencer Register

Offset: 314h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (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

Bit	Reset	Description
		4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

20.2.4.174 APBDMACHAN_CHANNEL_24_APB_PTR_0

APB-DMA-24 APB Starting Address Pointer Register

Offset: 318h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.175 APBDMACHAN_CHANNEL_24_APB_SEQ_0

APB-DMA-24 APB Address Sequencer Assignments

Offset: 31ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: when enabled the data going to 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 (def) 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 Window (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

20.2.4.176 APBDMACHAN_CHANNEL_25_CSR_0

APB-DMA-25 Control

Offset: 320h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enables DMA channel transfer 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
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 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3

Bit	Reset	Description
		18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.177 APBDMACHAN_CHANNEL_25_STA_0

APB-DMA-25 Status Register

Offset: 324h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activate 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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.178 APBDMACHAN_CHANNEL_25_DMA_BYTE_STA_0

APB-DMA-25 DMA Byte Status Register

Offset: 328h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.179 APBDMACHAN_CHANNEL_25_AHB_PTR_0

APB-DMA-25 AHB Starting Address Pointer Register

Offset: 330h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.180 APBDMACHAN_CHANNEL_25_AHB_SEQ_0

APB-DMA-25 AHB Address Sequencer Register

Offset: 334h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (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

20.2.4.181 APBDMACHAN_CHANNEL_25_APB_PTR_0

APB-DMA-25 APB Starting Address Pointer Register

Offset: 338h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.182 APBDMACHAN_CHANNEL_25_APB_SEQ_0

APB-DMA-25 APB Address Sequencer Assignments

Offset: 33ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to 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 (def) 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 Window (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

20.2.4.183 APBDMACHAN_CHANNEL_26_CSR_0

APB-DMA-26 Control

Offset: 340h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25

Bit	Reset	Description
		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
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 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.184 APBDMACHAN_CHANNEL_26_STA_0

APB-DMA-26 Status Register

Offset: 344h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activated 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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.185 APBDMACHAN_CHANNEL_26_DMA_BYTE_STA_0

APB-DMA-26 DMA Byte Status Register

Offset: 348h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.186 APBDMACHAN_CHANNEL_26_AHB_PTR_0

APB-DMA-26 AHB Starting Address Pointer Register

Offset: 350h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.187 APBDMACHAN_CHANNEL_26_AHB_SEQ_0

APB-DMA-26 AHB Address Sequencer Register

Offset: 354h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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	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 (4x32bits)else = 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 (def) (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

20.2.4.188 APBDMACHAN_CHANNEL_26_APB_PTR_0

APB-DMA-26 APB Starting Address Pointer Register

Offset: 358h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.189 APBDMACHAN_CHANNEL_26_APB_SEQ_0

APB-DMA-26 APB Address Sequencer Assignments

Offset: 35ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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

Bit	Reset	Description
27	0x0	APB_DATA_SWAP: when enabled the data going to 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 (def) 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 Window (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

20.2.4.190 APBDMACHAN_CHANNEL_27_CSR_0

APB-DMA-27 Control

Offset: 360h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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

Bit	Reset	Description
		18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
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 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.191 APBDMACHAN_CHANNEL_27_STA_0

APB-DMA-27 Status Register

Offset: 364h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activated or waiting 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

Bit	R/W	Reset	Description
			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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.192 APBDMACHAN_CHANNEL_27_DMA_BYTE_STA_0

APB-DMA-27 DMA Byte Status Register

Offset: 368h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.193 APBDMACHAN_CHANNEL_27_AHB_PTR_0

APB-DMA-27 AHB Starting Address Pointer Register

Offset: 370h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.194 APBDMACHAN_CHANNEL_27_AHB_SEQ_0

APB-DMA-27 AHB Address Sequencer Register

Offset: 374h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS

Bit	Reset	Description
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 (def) (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

20.2.4.195 APBDMACHAN_CHANNEL_27_APB_PTR_0

APB-DMA-27 APB Starting Address Pointer Register

Offset: 378h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.196 APBDMACHAN_CHANNEL_27_APB_SEQ_0

APB-DMA-27 APB Address Sequencer Assignments

Offset: 37ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to 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 (def) 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 Window (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

20.2.4.197 APBDMACHAN_CHANNEL_28_CSR_0

APB-DMA-28 Control

Offset: 380h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
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 = NA6

Bit	Reset	Description
		7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.198 APBDMACHAN_CHANNEL_28_STA_0

APB-DMA-28 Status Register

Offset: 384h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activated 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
15:2	RO	X	COUNT: Current 32-bit word cycles Flags set /cleared by HW

20.2.4.199 APBDMACHAN_CHANNEL_28_DMA_BYTE_STA_0

APB-DMA-28 DMA Byte Status Register

Offset: 388h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.200 APBDMACHAN_CHANNEL_28_AHB_PTR_0

APB-DMA-28 AHB Starting Address Pointer Register

Offset: 390h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.201 APBDMACHAN_CHANNEL_28_AHB_SEQ_0

APB-DMA-28 AHB Address Sequencer Register

Offset: 394h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (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

Bit	Reset	Description
		4 = WRAP_ON_256WORDS 5 = WRAP_ON_512WORDS 6 = WRAP_ON_1024WORDS 7 = WRAP_ON_2048WORDS

20.2.4.202 APBDMACHAN_CHANNEL_28_APB_PTR_0

APB-DMA-28 APB Starting Address Pointer Register

Offset: 398h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.203 APBDMACHAN_CHANNEL_28_APB_SEQ_0

APB-DMA-28 APB Address Sequencer Assignments

Offset: 39ch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to 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 (def) 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 Window (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

20.2.4.204 APBDMACHAN_CHANNEL_29_CSR_0

APB-DMA-29 Control

Offset: 3a0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	ENB: Enables DMA channel transfer 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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
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 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3

Bit	Reset	Description
		18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.205 APBDMACHAN_CHANNEL_29_STA_0

APB-DMA-29 Status Register

Offset: 3a4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activated 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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.206 APBDMACHAN_CHANNEL_29_DMA_BYTE_STA_0

APB-DMA-29 DMA Byte Status Register

Offset: 3a8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.207 APBDMACHAN_CHANNEL_29_AHB_PTR_0

APB-DMA-29 AHB Starting Address Pointer Register

Offset: 3b0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.208 APBDMACHAN_CHANNEL_29_AHB_SEQ_0

APB-DMA-29 AHB Address Sequencer Register

Offset: 3b4h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 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 (def) (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

20.2.4.209 APBDMACHAN_CHANNEL_29_APB_PTR_0

APB-DMA-29 APB Starting Address Pointer Register

Offset: 3b8h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.210 APBDMACHAN_CHANNEL_29_APB_SEQ_0

APB-DMA-29 APB Address Sequencer Assignments

Offset: 3bch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to 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 (def) 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 Window (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

20.2.4.211 APBDMACHAN_CHANNEL_30_CSR_0

APB-DMA-30 Control

Offset: 3c0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 2 = SMP25

Bit	Reset	Description
		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
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 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32-bit word cycles

20.2.4.212 APBDMACHAN_CHANNEL_30_STA_0

APB-DMA-30 Status Register

Offset: 3c4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activated 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
15:2	RO	X	COUNT: Current 32-bit word cycles Flags set /cleared by HW

20.2.4.213 APBDMACHAN_CHANNEL_30_DMA_BYTE_STA_0

APB-DMA-30 DMA Byte Status Register

Offset: 3c8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.214 APBDMACHAN_CHANNEL_30_AHB_PTR_0

APB-DMA-30 AHB Starting Address Pointer Register

Offset: 3d0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.215 APBDMACHAN_CHANNEL_30_AHB_SEQ_0

APB-DMA-30 AHB Address Sequencer Register

Offset: 3d4h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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	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 (4x32bits)else = 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 (def) (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

20.2.4.216 APBDMACHAN_CHANNEL_30_APB_PTR_0

APB-DMA-30 APB Starting Address Pointer Register

Offset: 3d8h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.217 APBDMACHAN_CHANNEL_30_APB_SEQ_0

APB-DMA-30 APB Address Sequencer Assignments

Offset: 3dch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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

Bit	Reset	Description
27	0x0	APB_DATA_SWAP: When enabled the data going to 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 (def) 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 Window (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

20.2.4.218 APBDMACHAN_CHANNEL_31_CSR_0

APB-DMA-31 Control

Offset: 3e0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 Transfer 0 = MULTIPLE_BLOCK 1 = SINGLE_BLOCK
26:22	0x0	TRIG_SEL: Enable on Non-Zero Value 0 = NA1 1 = SMP24 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

Bit	Reset	Description
		18 = APB_9 19 = APB_10 20 = APB_11 21 = APB_12 22 = APB_13 23 = APB_14 24 = APB_15
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 = NA6 7 = NA7 8 = UART_A 9 = UART_B 10 = UART_C 11 = DTV 12 = NA12 13 = NA13 14 = NA14 15 = SL2B1 16 = SL2B2 17 = SL2B3 18 = SL2B4 19 = UART_D 20 = UART_E 21 = I2C 22 = I2C2 23 = I2C3 24 = DVC_I2C 25 = OWR 26 = I2C4 27 = SL2B5 28 = SL2B6 29 = NA29 30 = NA30 31 = NA31
15:2	0x0	WCOUNT: Number of 32bit word cycles

20.2.4.219 APBDMACHAN_CHANNEL_31_STA_0

APB-DMA-31 Status Register

Offset: 3e4h | Read/Write: R/W | Reset: 0bx0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	BSY: indicate DMA Channel Status activated 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

Bit	R/W	Reset	Description
			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
15:2	RO	X	COUNT: Current 32bit word cycles Flags set /cleared by HW

20.2.4.220 APBDMACHAN_CHANNEL_31_DMA_BYTE_STA_0

APB-DMA-31 DMA Byte Status Register

Offset: 3e8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:0	X	DMA_COUNT: Indicates the actual DMA Data Transfer Count in bytes

20.2.4.221 APBDMACHAN_CHANNEL_31_AHB_PTR_0

APB-DMA-31 AHB Starting Address Pointer Register

Offset: 3f0h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	AHB_BASE: APB-DMA Starting Address for AHB Bus: SW writes to modify

20.2.4.222 APBDMACHAN_CHANNEL_31_AHB_SEQ_0

APB-DMA-31 AHB Address Sequencer Register

Offset: 3f4h | Read/Write: R/W | Reset: 0b00100110xxxx0000

Bit	Reset	Description
31	0x0	INTR_ENB: 0 = send interrupt to COP 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 (Def) 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 (4x32bits)else = 8 Words (8x32bits) default 4 = DMA_BURST_1WORDS 5 = DMA_BURST_4WORDS 6 = DMA_BURST_8WORDS

Bit	Reset	Description
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 (def) (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

20.2.4.223 APBDMACHAN_CHANNEL_31_APB_PTR_0

APB-DMA-31 APB Starting Address Pointer Register

Offset: 3f8h | Read/Write: R/W | Reset: 0b000000000000000000

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

20.2.4.224 APBDMACHAN_CHANNEL_31_APB_SEQ_0

APB-DMA-31 APB Address Sequencer Assignments

Offset: 3fch | Read/Write: R/W | Reset: 0b0100xxxxxxxx001

Bit	Reset	Description
30:28	0x2	APB_BUS_WIDTH: 0 = 8 bit Bus 1 = 16 bit Bus 2 = 32 bit Bus (Def) 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	APB_DATA_SWAP: When enabled the data going to 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 (def) 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 Window (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

21.0 USB COMPLEX

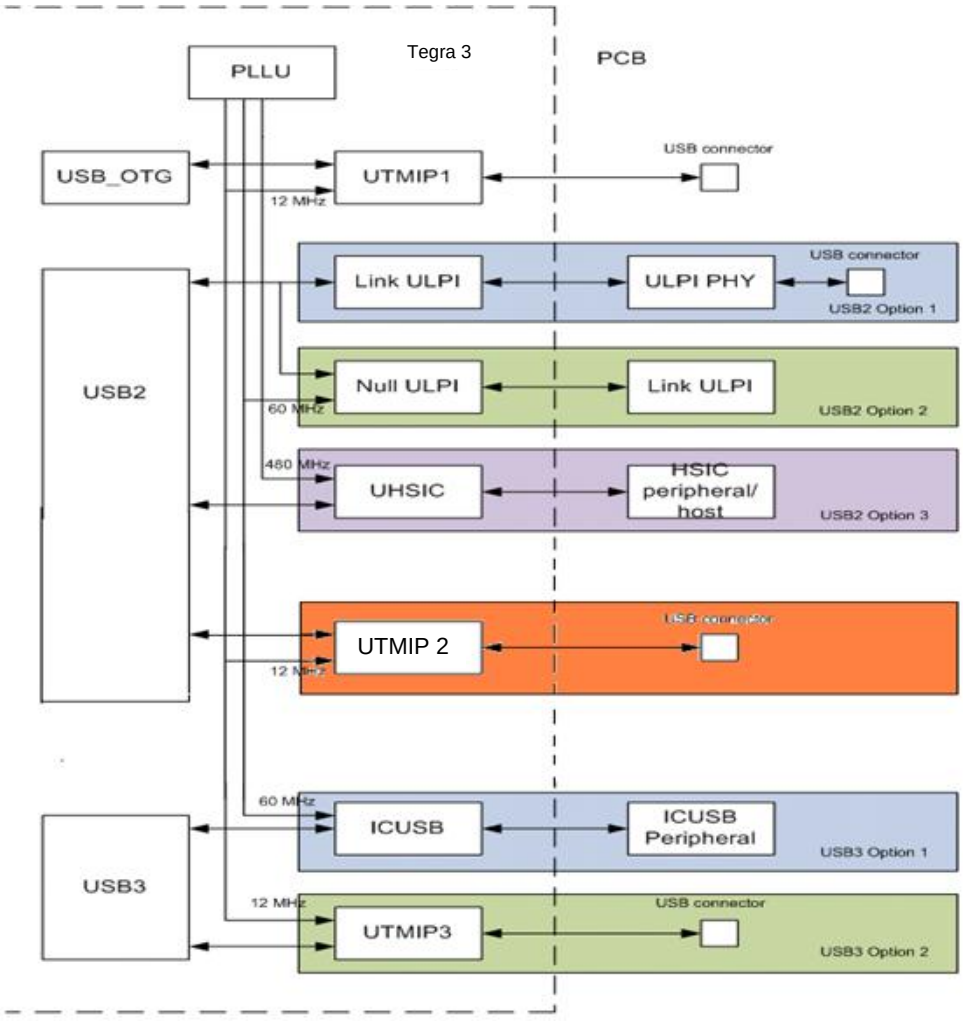
The USB complex in Tegra® 3 devices provides a mechanism to communicate with a PC and/or a peripheral such as a USB keyboard, mouse, camera, etc. using regular USB ports and to an on-board baseband controller using either USB/ULPI/HSIC/ICUSB interfaces. It consists of 3 USB controllers and 7 USB interfaces: UTMIP1, Link ULPI, Null ULPI, HSIC, UTMIP2, ICUSB and UTMIP3. Figure 36 shows the relationship between different USB controllers and interfaces.

Tegra 3 devices implement VBUS and ID sensors required for device/host functionality and associated interrupt mechanisms; software uses these for the device/host state machine in software. Battery charger detection is implemented as specified in USB Battery charger specification version 1.1; software needs to implement the charger sequence.

The USB controllers implement transaction level processing of control requests. Software decodes the type of control request and programs the packets appropriately. Suspend/resume functionality saves power at times when USB functionality is not required; there are automatic wake-up events and interrupts that software can program.

21.1 USB Controllers and Interfaces

Figure 36. Tegra 3 USB Controllers and Interfaces



USB Device/Host and UTMIP1 (USB1 Port)

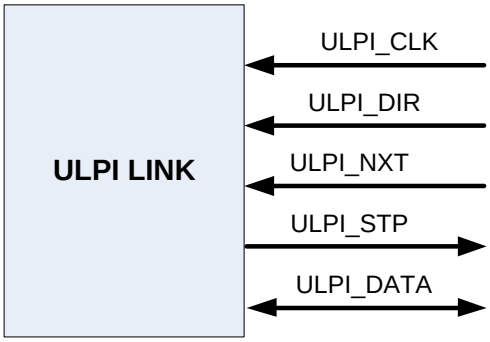
This is the primary USB port on Tegra 3 devices. It supports USB device and USB host modes. USB recovery is supported on this interface only. Wake-up from deep sleep mode is also supported on VBUS and accessory detect (ACCx_DETECT) pins. Power and Ground pins for USB1 and USB3 are shared.

USB2 and Link ULPI

This interface supports connecting an on-board external ULPI PHY to use as additional USB port. Depending on ULPI PHY capabilities, it can support USB Host and USB Peripheral modes of operation.

Link ULPI interface shares the pins with Null ULPI and other interfaces of Tegra 3 devices. It is required that the pin-mux be programmed prior to using the Link ULPI interface.

Link ULPI is a slave on the ULPI bus, in that the 60 MHz ULPI interface clock is driven from the external ULPI PHY on the ULPI_CLK pin. Note the change of direction in the pins ULPI_DIR, ULPI_NXT and ULPI_STP with respect to Null ULPI interface.

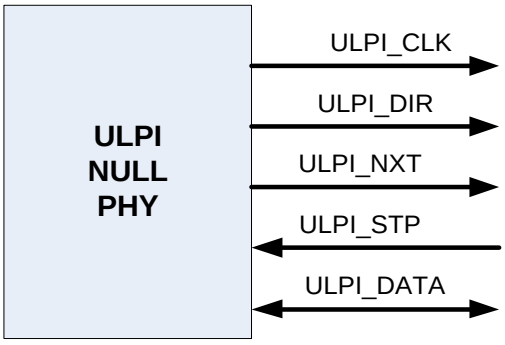


USB2 and Null ULPI

This interface allows Tegra 3 devices to behave like a ULPI PHY so that an external Link ULPI controller can be connected to it. This is used to support baseband controllers that use ULPI interface.

Null ULPI interface shares the pins with Link ULPI and other interfaces of Tegra 3 devices. It is required that the pin-mux be programmed prior to using the Null ULPI interface.

Null ULPI is a master on the ULPI bus, in that the 60 MHz ULPI interface clock is driven by Tegra 3 devices to the external Link on the ULPI_CLK pin. Note the change of direction in the pins ULPI_DIR, ULPI_NXT and ULPI_STP with respect to Link ULPI interface.



USB2 and UHSIC

This interface allows connection of an on-board HSIC Peripheral/Host to the Tegra 3 devices. This is used to support baseband controllers that use HSIC interface.

UHSIC supports both Host and Peripheral modes. It does not support switching between Host and Peripheral modes. Since the external HSIC Peripheral/Host is always on-board, the role of Peripheral and Host is expected to be fixed at board design time.

USB2 and UTMIP2

This interface allows an additional USB port. It can support native Host functionality only

USB3 and ICUSB

This interface allows connection of an on-board ICUSB Peripheral to the Tegra 3 devices. This is used to support baseband controllers that use ICUSB interface.

The ICUSB interface on Tegra 3 devices supports two power-supply voltages: 1.8V and 3.0V. Software needs to perform proper power sequencing as described in the ICUSB specification.

USB3 and UTMIP3 (USB3 port)

This is the third USB port on Tegra 3 devices. It supports Host a mode of operation only.

Power and Ground pins for USB1, USB2 and USB3 are shared.

21.1.1 Interface Restrictions

Link ULPI, Null ULPI and UHSIC are all exclusive interfaces on USB2 controller, and only one of the interfaces can be used in a given board design. Similarly, ICUSB and UTMIP3 are exclusive interfaces on USB3 controller, only one of the interfaces can be used in a given board design.

On power-on-reset, USB2 defaults to UTMIP mode, and USB3 defaults to ICUSB mode.

21.2 Controller

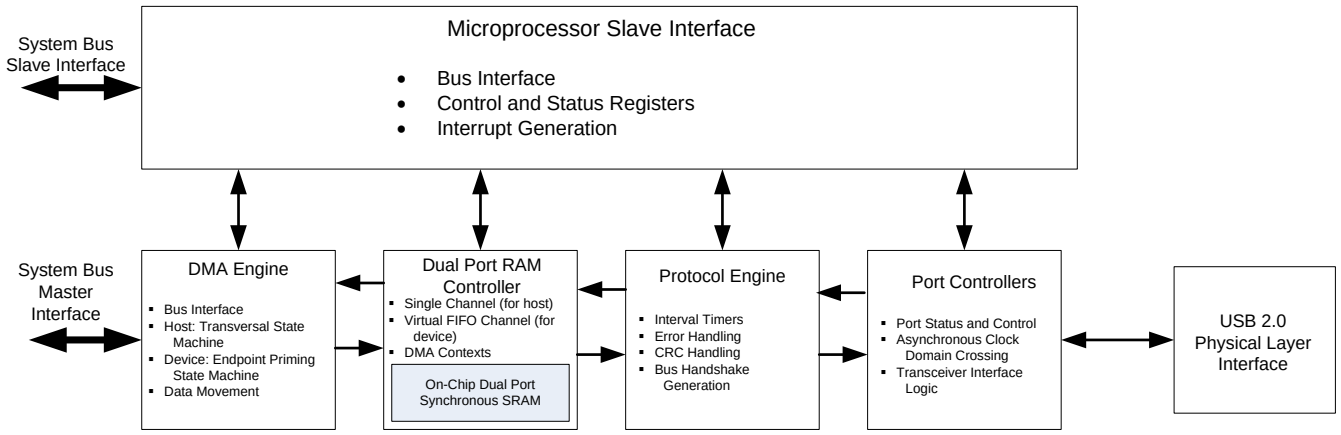
All three USB controllers offer the same functionality, although the interface may restrict any one to a subset of the possible functions.

The USB Controller can be either a USB host or device. It has a USB 2.0 High Speed dual role USB Host controller or USB Device controller.

- Device controller registers and data structures are implemented as extensions to the EHCI programmers interface.
- Supports USB legacy (USB 1.1) Full and Low speed devices without a companion USB 1.1 host controller or host controller driver software using EHCI standard data structures.

The block diagram of USB is shown in the figure below.

Figure 37. USB Controller Block Diagram



21.2.1 Endpoint Capabilities

All controllers support 16 bi-directional endpoints in device mode. Endpoint 0 is always a bi-directional control endpoint. All other endpoints can be programmed as one of Bulk, Interrupt, Isochronous or Control type in either direction. Control endpoints are always bi-directional 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 max packet size supported on any endpoint is 512 bytes in high-speed mode, for both device and host modes.

21.2.2 AHB Interface

All USB controllers use an AHB interface as master and slave for communicating with other parts of Tegra 3 devices. Table 33 lists the AHB interface numbers for each controller.

Table 33: AHB Master and Slave Numbers

Controller	AHB Master Number	AHB Slave Number
USB1	6	6
USB2	18	9
USB3	17	11

21.3 USB Programming Guidelines

21.3.1 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 and software should bring it out of reset by first setting SWR_USBD_RST in RST_DEVICES_L register to ENABLE and then set 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, PHY clock takes some time to come up, so software must wait until USB_PHY_CLK_VALID bit in USB_SUSP_CTRL register becomes valid. This bit, if set to 1, also generates an interrupt if RSM_IE is set in USB_SUSP_CTRL register.

21.3.1.1 ID and VBUS Connection

With Tegra 3 devices, you can use ACCx_DETECT supported by the UTMIP/USB hardware, or use a GPIO for ID functions. If using a GPIO as the ID, 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 **PHY_VBUS_WAKEUP_ID_0** register. It is recommended that board design connects ACCx_DETECT pin on the Tegra 3 devices to the ID pin on the micro-AB connector even if the ID is connected to a GPIO.

Tegra 3 devices have a VBUS pin connected to UTMIP and 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 Tegra 3' VBUS pin. It is required that its voltage is above 3V when VBUS is on.

Both VBUS and ACCx_DETECT 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.

21.3.1.2 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 3 devices because of the weak pull-up on this pin. This makes the Tegra 3 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 3 devices, 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 Tegra 3 devices. 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 places the Tegra 3 devices in host mode.
- User connects micro-B (or mini-B) end of the cable to Tegra 3 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 3 devices 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 PMC.

21.3.1.3 PHY Clock Control

The PHY clock can be turned off using the bit SUSP_SET in 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. Don't 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.

There is an interrupt generated whenever PHY clock is turned on which can be checked by checking the value of USB_PHY_CLK_VALID in USB_SUSP_CTRL register to be 1. The interrupt can be enabled/disabled by setting the bit RSM_IE 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 USB cable is not connected
- When the USB Host puts the USB bus in suspend mode

When USB cable isn't 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 the section on Detection of cable insertion.

When 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 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 WK_RSM_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 wakeup USB system, then it needs to turn on the PHY clock as described above.

21.3.1.4 Charger Detection 1 (Non-compliant Chargers)

There are 4 conditions for the different kind of non-compliant chargers: SE1, FS-PU, LS-PU, SE0. We can have some hooks in SW to support any kind of non-compliant charger by following the generic procedure below:

1. VBUS detection (by whatever means SW does it)
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 SW doesn't read wrong values.
5. Read the LS bits [11:10] in 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. If case 1, 2, or 3, enable charging current as per that charger requirement. This could be different for every implementation.
7. If case 4, either we are connected to PC host or a charger_4 (which could be USB compliant charger as well). Go to step 10.
8. We can set the USB circuit on Tegra 3 devices to low-power mode now. Make sure the charger circuit doesn't change state.
9. VBUS disconnect. Go back to step 1.
10. Check if 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 next step.
11. Try to set USB controller to device mode and enable by setting it to RUN mode. Wait for SOF.
12. If SOF is not seen after timeout, then we are connected to charger_4. Set current limit as required for that charger. Go to Step 7.

21.3.1.5 Charger Detection 2 (Compliant Chargers)

This substitutes step 10 from 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.
2. Set the other register bits in UTMIP_BAT_CHRG_CFG0 register to the following value:
    - UTMIP_OP_I_SRC_EN = 1
3. Wait for about 10 microseconds to allow the battery charger detector to settle.
4. 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 we are connected to a USB compliant charger.
5. Set the registers bits in UTMIP_BAT_CHRG_CFG0 register to the following values:
    - UTMIP_OP_I_SRC_EN = 0
6. 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
7. Wait for about 10 microseconds to allow the battery charger detector to settle.

8. 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 we are connected to a USB compliant charger.
    - If it isn't 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.

21.3.2 USB2 Programming Sequence

USB2 controller supports Link ULPI, Null ULPI and UHSIC interfaces. All of these interfaces are directly controlled through USB2 register space.

21.3.2.1 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 SWR_USB2_RST in RST_DEVICES_H register to ENABLE and then set it to DISABLE.

After the clocks for USB2 are up, SW can program any registers for USB2 required for proper configuration.

PLLU outputs a 60 MHz clock FO_ICUSB for Null ULPI interface and a 480 MHz clock FO_UHSIC for UHSIC interface. If using any of these interfaces, PLLU should be programmed appropriately.

Any time after USB2 is reset, or ULPI/UHSIC PHY comes out of suspend, software needs to wait until ULPI/UHSIC PHY clock comes up. It can do that by checking for USB_PHY_CLK_VALID bit in USB2_IF_USB_SUSP_CTRL register. If this bit is 1, PHY clocks are up, else they aren't. 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.

21.3.2.2 Selection of Link ULPI interface

If using Link or Null ULPI interface, software can hold UHSIC in reset by setting UHSIC_RESET in USB2_IF_USB_SUSP_CTRL register to 1. Similarly hold UTMIP in reset by setting UTMIP_RESET IN USB2_IF_USB_SUSP_CTRL register to 1.

1. Disable pull-ups on ULPI data pins by setting PULL_UP fields in **PINMUX_AUX_ULPI_DATA*_0** to NORMAL (2'b00)
2. Enable ULPI interface by writing 1 to ULPI_PHY_ENB bit in USB2_IF_USB_SUSP_CTRL register.
3. Bypass the pin-mux on ULPI outputs by writing 1 to ULPI_OUTPUT_PINMUX_BYP and ULPI_CLKOUT_PINMUX_BYP fields in USB2_IF_ULPI_TIMING_CTRL_0 register.
4. Remove the tri-state on ULPI pins by setting TRISTATE field in **PINMUX_AUX_ULPI_*_0** registers NORMAL (1'b0).

21.3.2.3 Selection of Null ULPI interface

If using Link or Null ULPI interface, software can hold UHSIC in reset by setting UHSIC_RESET in USB2_IF_USB_SUSP_CTRL register to 1.

1. Program the PLLU parameters.
2. Disable pull-ups on ULPI pins by setting PULL_UP field in **PINMUX_AUX_ULPI_*_0 registers** to NORMAL (2'b00).
3. Enable ULPI interface by writing 1 to ULPI_PHY_ENB bit in USB2_IF_USB_SUSP_CTRL register.
4. Set the ULPI pin-mux settings by writing 1 to the fields ULPI_OUTPUT_PINMUX_BYP, ULPI_CLKOUT_PINMUX_BYP, ULPI_CLK_OUT_ENA, ULPI_CLK_PADOUT_ENA, ULPI_SHADOW_CLK_SEL, and ULPI_CORE_CLK_SEL in USB2_IF_ULPI_TIMING_CTRL_0 register.
5. Enable PLLU 60 MHz output by writing 1 to ULPIS2S_PLLU_MASTER_BLASTER60 field in USB2_IF_USB_ULPIS2S_CTRL register.
6. Remove the tri-state on ULPI pins by setting TRISTATE field in **PINMUX_AUX_ULPI_*_0** registers to NORMAL (1'b0).

21.3.2.4 Selection of UHSIC interface

UHSIC IO 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.

1. Hold UHSIC In reset by writing 1 to UHSIC_RESET field in USB2_IF_USB_SUSP_CTRL register.
2. Program the PLLU parameters to enable UHSIC PLLU clock.
3. Select UHSIC interface by writing 1 to UHSIC_PHY_ENB field in USB2_IF_USB_SUSP_CTRL register.
4. Program the configuration parameters for UHSIC.
5. Release reset to UHSIC by writing 0 to UHSIC_RESET field in USB2_IF_USB_SUSP_CTRL register.
6. Set CM field in USB2D_USBMODE register to HOST mode.
7. Program the USB2 controller to use UHSIC PHY by writing 0 to PTS field in USB2_CONTROLLER_1_USB2D_HOSTPC1_DEVLC register.
8. Program the UHSIC_HS_POSTAMBLE_OUTPUT_ENABLE field in the UHSIC_TX_CFG0 register to "1".
9. Wait until PHY clock comes up by checking for USB_PHY_CLK_VALID bit in USB2_IF_USB_SUSP_CTRL register. Interrupt can be generated as explained above.

There is no VBUS or ID detection required for this interface as these pins don't exist for UHSIC.

21.3.2.5 Selection of UTMIP Interface

UTMIP IO pads are in power-down state at power-on. Bring them out of power-down mode by writing the following fields to 0:

1. FORCE_PD_POWERDOWN, FORCE_PD2_POWERDOWN, FORCE_PDZI_POWERDOWN fields in UTMIP_XCVR_CFG0 register.
2. OTGPD and BIASPD fields in UTMIP_BIAS_CFG0 register.
3. FORCE_PDDISC_POWERDOWEN , FORCE_PDCHRP_POWERDOWN, FORCE_PDDR_POWERDOWN field in UTMIP_XCVR_CFG1 register.
4. Hold UTMIP PHY in reset by writing UTMIP_RESET bit in USB2_IF_USB_SUSP_CTRL register to 1.
5. Program the PLLU parameters to enable UTMIP PLLU clock.
6. Bring the corresponding data sampler B out of power down (see CLK_RST_CONTROLLER_UTMIP_PLL_CFG2_0 register below)
7. Enable UTMIP interface by setting UTMIP_PHY_ENB in USB2_IF_USB_SUSP_CTRL register to 1.
8. Program the configuration parameters for UTMIP.

9. Release reset to UTMIP by writing 0 to UTMIP_RESET field in USB2_IF_USB_SUSP_CTRL register.
10. Wait until PHY clock comes up by checking for USB_PHY_CLK_VALID bit in USB2_IF_USB_SUSP_CTRL register. Interrupt can be generated as explained above.
11. Program the USB controller to use UTMIP PHY by setting the PTS field in USB2_CONTROLLER_1_USB2D_HOSTPC1_DEVLC register to UTMIP (2'b00).

21.3.2.6 ID and VBUS detection in Link ULPI mode

For external ULPI PHY, the ID and VBUS can be connected to the USB connector as per the guidelines given by ULPI PHY vendor. The detection for ID and VBUS can be done in two ways:

- USB2 controller registers USB2_CONTROLLER_USB2D_OTGSC can be used to enable ID_PU (ID pull-up), and read status on VBUS and ID sensors.
- Use the USB2D_ULPI_VIEWPORT register for USB2 to perform writes/reads to the external ULPI PHY.

Setting the PP bit in USB2D_PORTSC1 register would enable the VBUS going out to the USB connector when USB2 is configured in Host mode. The board designer needs to connect appropriate signals from the ULPI PHY in order to make correct use of this feature.

Alternately, software can write appropriate register in ULPI PHY according to the datasheet of the particular PHY using the USB2D_ULPI_VIEWPORT register.

Note that USB2 is disabled during DPD (LP0) mode. If system designer wants to wake up the system on a cable connection for USB2, then ID and VBUS should be brought to PMU and PMU can be programmed to wake up the Tegra 3 devices on an event on these pins. Alternately, these can be driven to GPIOs and those GPIOs can be used in PMC as wakeup sources.

For details on USB2D_ULPI_VIEWPORT, please see the description for that register in the register section.

21.3.2.7 Detection of USB Cable Insertion in Link ULPI mode

The USB cable insertion event can be detected by a change in the ID or VBUS pin on the micro-AB connector connected to ULPI PHY. Initially, when the cable is not attached, the ID pin on the micro-AB connector is floating and is detected as 1 by Tegra 3 devices because of the weak pull-up on this pin. This makes the Tegra 3 devices USB2 start-up as a B-device, even though a cable is not connected (When the ID pin is seen as a 1 by Tegra 3 devices, there can be no-cable or micro-B connected).

When a user attaches the USB cable to the ULPI PHY port, one of the following two events occur:

- User connects micro-A end of the cable to Tegra 3 devices. This changes the ID pin to 0. Hence the software can detect the ID change by reading the USB2D_OTGSC register and go to A-device mode. Software can then supply power to VBUS on ULPI PHY port and place the Tegra 3 device USB2 in host mode.
- User connects micro-B (or mini-B) end of the cable to Tegra 3 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_Vbus_Vld level using the USB2D_OTGSC register. Once this is seen, it can place the Tegra 3 device USB2 in the device mode.

21.3.2.8 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 USB1 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. Don't 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.

There is an interrupt generated whenever PHY is woken up from suspend which can be checked by checking the value of USB_WAKEUP_INT_STS in USB_SUSP_CTRL register to be 1. The interrupt can be enabled/disabled by setting the bit USB_WAKEUP_INT_ENB in USB_SUSP_CTRL to 1/0.

There is an interrupt generated whenever PHY clock is turned on which can be checked by checking the value of USB_PHY_CLK_VALID_INT_STS in USB_SUSP_CTRL register to be 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 waked 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:

- This suspend is not supported in null ULPI mode.
- When doing this suspend for link ULPI mode,

a. USB2 clock should be enabled (CLK_ENB_USBD in ARCLK_RST_CLK_OUT_ENB_L register should be set to ENABLE).

b. USB2 reset should be disabled (SWR_USB2_RST in RST_DEVICES_H register should be set to DISABLE).

Not doing this could lead to undesirable side-effects.

- When USB2 is in suspend, AHB clocks to USB2 controller is turned off, and hence there should be no register access to USB2_CONTROLLER_1_* registers. All registers marked as ARUSB2_IF_* are accessible, and can be used to resume the ULPI PHY clocks.

There are two cases to consider for controlling the ULPI PHY clocks.

Device Mode

The PHY clock can be turned off in two cases:

- When USB cable isn't connected
- When the USB Host puts the USB bus in suspend mode

When USB cable isn't 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 the section on Detection of cable insertion.

When 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 USB2_IF_USB_SUSP_CTRL register to turn on the PHY clock on a USB resume or a USB reset event from USB Host: USB_WAKE_ON_DISCON_EN_DEV: Wake on Disconnect enable in device mode, and USB_WAKE_ON_RESUME_EN - Wake on Resume enable.

If USB_WAKE_ON_RESUME_EN in USB2_IF_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 USB_WAKE_ON_DISCON_EN_DEV in USB2_IF_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 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 wakeup USB system, then it needs to turn on the PHY clock as described above.

Separate resets added for Slv0, Slv1, line, pads. Default resets on these need to be removed. 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 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

21.3.2.9 Bus Signaling Procedure changes for 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 sections 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:

1. Check if “LineState” (field LS in USB2D_PORTSC1 register) is DPlus – Make sure the HSIC bus is IDLE before starting the connect;
2. Program the XCVR to detect SHORT connect by writing “1” to DETECT_SHORT_CONNECT field in the UHSIC_MISC_CFG0 register.
3. Program the FORCE_XCVR_MODE field in UHSIC_MISC_CFG0 register to “1”.
4. Poll until “Device Connect” is received by polling until the UHSIC_CONNECT_DETECT field in the UHSIC_STAT_CFG0 register becomes “1”.

5. Check “PortSpeed” (field PSPD in USB2D_PORTSC1 register) – The ultimate check is to verify the bus connection speed as reported by the USB2 controller. It should match the HSIC native speed (High Speed).
6. The HSIC state machine does not detect a CONNECT if PHY clock is disabled. So SW should take care of this design limitation and should **NOT**
   - Either turn off PHY clock to detect the connect event. Or,
   - Expect a connect event if PHY clock in OFF.
7. The HSIC controller FSM detects CONNECT even before the actual CONNECT happens on the HSIC interface. This is due to the fact that FSM assumes CONNECT as soon as PHY type is selected to HSIC. From a controller perspective this makes sense as HSIC device only get connected once per power up event and there is no disconnect. Based on this the controller assumes that HSIC device is always connected. Since HSIC connect event doesn’t easily translate into linestate, this reduces dependency on third-party HSIC PHY designs.

Host Mode: Bus Reset Sequence

This is not the standard connect but rather a re-connect 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 3 HSIC as native HSIC support in-built.

Device Mode: Bus Connect Sequence

Generally (assuming all initial procures have already been run), a connect will occur when both the RUNSTOP bit in USB2D_USBCMD register has been enabled and the Host has enabled the port power on its side. There is no physical connection event when dealing with a HSIC bus and this signaling mechanism guarantees a successful connect whichever component reaches the respective connection point first.

In a standard connection, software would wait for a “bus reset” System Interrupt to start the speed negotiation procedure. With a HSIC bus instead, reset does not take place and software must drive the USB2 controller into the “HS Connected Idle” state:

1. Enable the RUNSTOP bit in register USB2D_USBCMD – The HSIC PHY will detect this and wait for a bus IDLE meaning the bus has connected. When that happens, it will show LineState as DPlus and place the bus in the CONNECT state for at least two strobe periods;
2. Poll LineState until it reaches DPlus – This is needed because the Host may not be ready when the device starts the connect. We can only proceed to the “HS Connected Idle” state when the HOST has seen the connect;
3. Follow the exact instructions from step 3 of the Host “Bus Connect Sequence”.
4. Enable the Port Force Full Speed Connect (field PFSC in USB2D_PORTSC1 register) – This is necessary if system implementation is to account for bus resets during operation. Mostly this will be the case therefore the feature is

mandatory. Otherwise a bus reset would drive the device USB2 controller into speed negotiation which is not allowed for an HSIC bus.

Device Mode: Bus Reset Sequence

A bus reset is always started by a Host. When this happens the USB2 controller Device will detect the reset and issue a system interrupt. Both the fields PR in USB2D_PORTSC1 register and URI in USB2D_USBSTS register are set and after the typical setup operations, in order to connect, software must also do:

1. Check if LineState is SE0 – If the Host has actually started to drive reset signaling on the bus, then the LineState should be SE0, otherwise the reset condition is false;
2. Poll until LineState is DPlus – This will make software execution synchronized with the end of the bus reset. After the bus reset and because of the “Force Full Speed” feature, the USB2 controller will be in “Full Speed Idle” state;
3. Follow step 3 of the Device “Bus Connect Sequence” – The procedure to follow is the same as above. Software must also enable the “Force Full Speed” feature at the end (as with step 4).

Note: During the above described port connection procedures, due to the direct driving of the USB2 controller through a few of its internal states, some hardware interrupts will be skipped (this is why some steps involve polling cycles). Because of this, and depending on the programmer's choice, software may simulate/ignore some system interrupts (making interrupt routines to be called/attended from within the same execution thread) to maintain compatibility with the rest of the stack. This may apply for example to port change interrupts at the end of the bus reset for both Host/Device modes.

21.3.3 USB3 Programming Sequence

USB3 controller supports ICUSB and UTMIP3 interfaces. All of these interfaces are directly controlled through USB3 register space.

21.3.3.1 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, SW 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 aren't. 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.

21.3.3.2 Selection of ICUSB Interface

If using ICUSB interface, software can hold UTMIP3 in reset by setting UTMIP3_RESET in USB3_IF_USB_SUSP_CTRL register to 1.

ICUSB interface only supports host mode of operation – a peripheral can be connected to Tegra 3 device’s ICUSB port. Peripheral mode of operation isn’t supported on ICUSB interface.

ICUSB IO pad is in power-down state at power-on. Bring it out of power-down mode by writing PD_ZI, PD_DR, PD_TX and IDDQ fields in ICUSB_XCVR_CFG register to 0.

1. Program slew rate to 0x2e by writing ICUSB_SLEW field into ICUSB_XCVR_CFG register which works across all PVT.
2. Program the PLLU parameters to enable ICUSB PLLU clock.
3. Select ICUSB interface by writing 1 to ICUSB_PHY_ENB field in USB3_IF_USB_SUSP_CTRL register.
4. Enable ICUSB module clock by writing 1 to ICUSB_MOD_CLK_ENB field in USB3_IF_USB_SUSP_CTRL register.
5. Wait until PHY clock comes up by checking for USB_PHY_CLK_VALID bit in USB2_IF_USB_SUSP_CTRL register. Interrupt can be generated as explained above.
6. Set CM field in USB2D_USBMODE register to HOST mode.
7. Change USB3 controller to ICUSB mode by writing iCUSB_SER to PTS field in USB2_CONTROLLER_2_USB2D_HOSTPC1_DEVLC register.
8. Enable appropriate voltage for ICUSB_VDD (1.8V or 3.0V) from PMU. Consult with the PMU vendor/datasheet about the programming required for this.
9. Enable ICUSB port in USB3 controller by writing 1 to IC_ENB1 field and appropriate ICUSB_VDD voltage in IC_VDD1 field of USB2D_ICUSB_CTRL register. For ICUSB_VDD of 1.8V, program 4 to IC_VDD1 and for ICUSB_VDD of 3.0V, program 5 to IC_VDD1. The value programmed in IC_VDD1 must match the voltage programmed in PMU for ICUSB_VDD.

Once initialization is done, standard USB drivers can be used to support ICUSB interface.

21.3.3.3 Selection of UTMIP3 interface

1. UTMIP3 IO pads are in power-down state at power-on. Bring them out of power-down mode by writing the following fields to 0:
   - FORCE_PD_POWERDOWN, FORCE_PD2_POWERDOWN, FORCE_PDZI_POWERDOWN fields in UTMIP_XCVR_CFG0 register.
   - OTGPD and BIASPD fields in UTMIP_BIAS_CFG0 register.
   - FORCE_PDDISC_POWERDOWEN , FORCE_PDCHRP_POWERDOWN, FORCE_PDDR_POWERDOWN field in UTMIP_XCVR_CFG1 register.
2. Hold UTMIP3 PHY in reset by writing UTMIP_RESET bit in USB3_IF_USB_SUSP_CTRL register to 1.
3. Program the PLLU parameters to enable UTMIP3 PLLU clock.
4. Enable UTMIP3 interface by setting UTMIP_PHY_ENB in USB3_IF_USB_SUSP_CTRL register to 1.
5. Program the configuration parameters for UTMIP3.
6. Bring the corresponding data sampler C out of power down (see CLK_RST_CONTROLLER_UTMIP_PLL_CFG2_0 register below)
7. Release reset to UTMIP3 by writing 0 to UTMIP_RESET field in USB3_IF_USB_SUSP_CTRL register.
8. 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.
9. Set CM field in USB2D_USBMODE register to HOST mode.
10. 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).

21.3.3.4 ID and VBUS Connection

UTMIP3 is primarily meant to support additional USB Host functionality on Tegra 3 devices. Hence for UTMIP3, ID and VBUS pins are not supported. GPIOs could be used in applications that need these pins. In such cases, software would set

appropriate *_SW_VALUE and *_SW_EN bits for each VBUS and ID sensors depending on the value detected from the GPIOs.

Both VBUS and ID are available as a wake-up event during DPD/LP0 state. Please consult PMC programming guidelines for this.

21.3.3.5 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 3 devices because of the weak pull-up on this pin. This makes Tegra 3 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 3 devices, 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 Tegra 3 devices. 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 3 device in host mode.
- User connects micro-B (or mini-B) end of the cable to Tegra 3 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 VBUS status on corresponding GPIO. Once this is seen, it can place the Tegra 3 devices in the device mode.

Note:	Both ID and VBUS change 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 PMC.

21.3.3.6 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 USB1 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. Don't 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.

There is an interrupt generated whenever PHY is waked up from suspend which can be checked by checking the value of USB_WAKEUP_INT_STS in USB_SUSP_CTRL register to be 1. The interrupt can be enabled/disabled by setting the bit USB_WAKEUP_INT_ENB in USB_SUSP_CTRL to 1/0.

There is an interrupt generated whenever PHY clock is turned on which can be checked by checking the value of USB_PHY_CLK_VALID_INT_STS in USB_SUSP_CTRL register to be 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 waked 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 is 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.

There are two cases to consider for controlling the PHY clocks.

Device Mode

The PHY clock can be turned off in two cases:

- When USB cable isn't connected
- When the USB Host puts the USB bus in suspend mode

When USB cable isn't 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 the section on Detection of cable insertion.

When 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_DEV: Wake on Disconnect enable during device mode, and USB_WAKE_ON_RESUME_EN - Wake on Resume enable.

If USB_WAKE_ON_RESUME_EN bit 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_DEV bit 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 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 wakeup USB system, then it needs to turn on the PHY clock as described above.

21.3.4 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. Assert and de-assert SWR_USB3_RST in RST_DEVICES_H register to bring the USB block out of reset.
3. Set 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.
   - Wait until PHY_CLKVALID is set to 1.
   - Clear the PD bits that are required for the normal operation.
10. From this on, 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.
11. If a cable disconnect is detected (VBUS = 0) or (ID = 1), then go back to step 4.

21.3.5 Recommended PHY Settings

- Set UTMIP_XCVR_SETUP to 0x5 and UTMIP_XCVR_SETUP_MSB to 0x3 of UTMIP_XCVR_CFG0_0 register. This helps in eye diagram.
- Set UTMIP_TERM_RANGE_ADJ to 0x7 of UTMIP_XCVR_CFG1 register. This helps in eye diagram.
- Set UTMIP_XCVR_HSSLEW_MSB to 0x8. This helps in eye diagram.

- Set disconnect level to 0x5. This can be done by setting UTMIP_HSDISCON_LEVEL to 0x1 and UTMIP_HSDISCON_LEVEL_MSB to 0x1 of UTMIP_BIAS_CFG0_0 register.
- Set squelch level to 0x2. This can be done by setting UTMIP_HSSQUELCH_LEVEL to 2'b10 UTMIP_BIAS_CFG0_0 register.
- Set HS SLEW to 2.92pf. This can be done by setting UTMIP_XCVR_HSSLEW_MSB to 7'h8, UTMIP_XCVR_HSSLEW to 2'b0 of UTMIP_XCVR_CFG0 register.

21.3.6 BOOTROM Initialization Sequence for USB Recovery

1. Program PLL_U.
   - Setting the PLLU_BASE register fields, PLLU_DIVM, PLLU_DIVN, PLLU_VCO_FREQ, PLLU_BYPASS and PLLU_ENABLE fields as described in the document.
2. Configuring USB
   - Bring up USB1 clocks by writing 1 to CLK_ENB_USBD in CLK_OUT_ENB_L register.
   - Assert and de-assert master USBD reset in CAR (SWR_USBD_RST in RST_DEVICES_L register) to bring USB1 out of reset.
   - 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 changing to 0.
   - To Use the A Session Valid for cable detection logic, set USB1_VBUS_SENSE_CTL field in USB1_LEGACY_CTRL register to A_SESS_VLD (2'b11).
   - 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.
   - Programming the tracking duration.
   - Setting USB1_IF_UTMIP_BIAS_CFG1.UTMIP_BIAS_PDTRK_COUNT as per the values given in the document.
   - Program the debouncer length times.
   - Setting UTMIP_DEBOUNCE_CFG0.UTMIP_BIAS_DEBOUNCE_A field.
   - 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 3rd 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 doc.
   - Set UTMIP_HSRX_CFG0.UTMIP_ELASTIC_LIMIT to 16.
   - Set UTMIP_HSRX_CFG1.UTMIP_HS_SYNC_START_DLY to 9
   - Resuscitate the crystal clock by setting UTMIP_PHY_XTAL_CLOCKEN in UTMIP_MISC_CFG1 register to 1 for USB.
3. Waiting for cable connect on USB UTMIP1 port. When cable is connected on USB UTMIP1 port,
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.
   - Do Bus reset.

- Wait until bus come out of reset.
- Set the controller in device mode.
- USB operations.

21.3.7 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 these programming need to be done before setting RS bit in USB2D_USBCMD to RUN (1) for the corresponding USB controller.

21.4 UTMIP Programming Guidelines

Tegra 3 devices have a total of 3 UTMI+ interfaces. This results in three 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. All UTMI+ PHY interfaces are identical instances of the same design. All three controllers use latest edition of USB core controller.

21.4.1 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.

21.4.2 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 (ULPI (null mode), IC USB, UTMIP, UHSIC). Table 34 describes the parameter settings that are dependent on the crystal clock reference frequency. The parameters are specified in decimal notation.

Table 34 480MHz 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. Please also note that PLLU_VCO_FREQ parameter must 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.

21.4.3 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 as it cannot do so rapidly enough to meet the protocol.

Table 35 PLL Automated Start Times (Must be setup in the Boot ROM)

CRYSTAL FREQ.	12.0 MHz	13.0 MHz	16.8 MHZ	19.2 MHz	26.0 MHz	38.4 MHz	48.0 MHz
PLLU_ENALBLE_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

The PLLU_ENABLE_DLY_COUNT should be for at least 1 microsecond, the PLLU_STABLE_COUNT should be for at least 1ms, the PLL_ACTIVE_DLY_COUNT for at least 10 microseconds, and the XTAL_FREQ_COUNT for about 2.5ms.

21.4.4 Fuse Programming

Tegra 3 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.

21.4.5 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 has been determined that 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 36 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[X]	Bias	UTMIP_OTGPD=1
ID_PD[X]	Bias	UTMIP_IDPD_SEL=1, UTMIP_IDPD_VAL=1
VBUS_WAKEUP_PD[X]	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_X	PHY_PLL	UTMIP_FORCE_PD_SAMP_X_POWERDOWN=1
ENABLE	PLL U	UTMIP_FORCE_PLLU_POWERDOWN=1
UTMIP	PHY	UTMIP_PHY_XTAL_CLOCKEN=0

The set of bits exists for both ports X, where X is 0, 1 or 2. Three of the power downs are explicitly labeled with a port index because they belong to the bias cell and happen to have a per port control. Otherwise the same register bits exist in both controllers.

21.4.6 UTMIP BIAS PAD Configurations

UTMIP BIAS pad is shared across all 3 USB controllers and 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

The sequence to configure the 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.

21.4.7 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.

21.4.8 Deep Power Down Behavior

Tegra 3 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. Each USB port 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 both USB ports are dedicated host ports 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.

21.4.9 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.

21.4.10 Static Boot ROM configuration for UTMIP

The boot ROM configuration for UTMIP should be done while the unit is held in Reset. There is a change in the PowerDown control of the various units, so it is a **MUST** that these are removed appropriately to see correct behavior. The following is the sequence for this:

1. Reset is applied to USB controller (and UTMIP).
2. Crystal clock is running for approximately 5 microseconds while UTMIP in reset.
3. Stop 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 section PLL_U and USB_PHY_PLL automatic startup times.
6. Remove power downs from USB_PHY_PLL ACTIVE/ENABLE/PD_SAMP_A and PLL_U.
7. Program the tracking duration as described in section Programming the Tracking Length Time. Remove the power downs from PD (Bias) and disable PD_TRK. After 25us (tracking time), enable PD_TRK, by writing 0 into UTMIP_FORCE_PDTRK_POWER_DOWN.
8. Once PD (Bias) is disabled, disable OTG_PD[X] after 1us.
9. Remove powerdowns from ID_PD[X] and VBUS_WAKEUP_PD[X]. It can be removed earlier too, as it has 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 400ns 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 the section Extending Debouncer Period Length.
13. Program various static parameters of the UTMIP as described in the section Miscellaneous Boot ROM fields.
14. Resuscitate 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.

21.4.11 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 10ms.
- 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 msec instead of 50 msec.

Port Suspend (PORTSC.Suspend) bit is set immediately after setting the bit. As per EHCI a 4msec delay is expected for this bit to be set.

Tegra 3 also has following areas 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

21.5 USB Registers

21.5.1 USB 1 Controller Registers

21.5.1.1 USB2_CONTROLLER_USB2D_ID_0

USB2D Identification Register

Offset: 000h | 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: Ones complement version of ID. This field is set to 0xF9.
7:0	X	ID: Configuration number. This field is set to 0x06

21.5.1.2 USB2_CONTROLLER_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 008h | Read/Write: RO | Reset: 0bxxxx

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.

21.5.1.3 USB2_CONTROLLER_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

Offset: 00ch | Read/Write: RO | Reset: 0bxxxxxx

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: No. of endpoints supported by this device controller. Set to 16. This includes control endpoint 0.

Bit	Reset	Description
0	X	DC: Device capable: Set to 1 indicating support for device mode.

21.5.1.4 USB2_CONTROLLER_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 010h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total no. 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 no. 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.

21.5.1.5 USB2_CONTROLLER_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 014h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total no. 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.

21.5.1.6 USB2_CONTROLLER_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

Offset: 100h | Read/Write: RO | Reset: 0bxxxxxxxx

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.

21.5.1.7 USB2_CONTROLLER_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 102h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

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.

21.5.1.8 USB2_CONTROLLER_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

Offset: 104h | 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.

21.5.1.9 USB2_CONTROLLER_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

Offset: 108h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxx

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 with 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 with 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".

21.5.1.10 USB2_CONTROLLER_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

Offset: 120h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

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.

21.5.1.11 USB2_CONTROLLER_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 124h | Read/Write: RO | Reset: 0bxxxxxxxxx

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 with 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.

21.5.1.12 USB2_CONTROLLER_USB2D_EXTSTS_0

USB2D EXTSTS Register

Offset: 128h | Read/Write: R/W | Reset: 0b0000x

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.

Bit	R/W	Reset	Description
			0 = DISABLE 1 = ENABLE

21.5.1.13 USB2_CONTROLLER_USB2D_USBEXTINTR_0

USB2D EXTINTR Register

Offset: 12ch | Read/Write: R/W | Reset: 0b00000

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
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

21.5.1.14 USB2_CONTROLLER_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 130h | Read/Write: R/W | Reset: 0b000000001000000x1x11x0000000

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 BA field of 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 a 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 4 = TWO_MF

Bit	R/W	Reset	Description
			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) 010 = 256 elements (1024 bytes) 011 = 128 elements

Bit	R/W	Reset	Description
			(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

21.5.1.15 USB2_CONTROLLER_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 134h | Read/Write: R/W | Reset: 0b00000000000000000x1x0x0000x0000

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 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

Bit	R/W	Reset	Description
			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
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 a 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 125us 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 125us 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
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

Bit	R/W	Reset	Description
			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

21.5.1.16 USB2_CONTROLLER_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 138h | Read/Write: R/W | Reset: 0b0000000000000000xxxxx0x000000000

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.
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 SW 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 SW 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

Bit	Reset	Description
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

21.5.1.17 USB2_CONTROLLER_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 13ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

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.

21.5.1.18 USB2_CONTROLLER_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 144h | Read/Write: R/W | Reset: 0b00000000000000000000

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 can not 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

Bit	Reset	Description
		bits correspond to memory address signals [31:12], respectively. Only used by the host controller.

21.5.1.19 USB2_CONTROLLER_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 148h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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.

21.5.1.20 USB2_CONTROLLER_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 14ch | Read/Write: R/W | Reset: 0b0x

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.

21.5.1.21 USB2_CONTROLLER_USB2D_BURSTSIZE_0

USB2D Burst Size register

Offset: 150h | Read/Write: R/W | Reset: 0b0000100000001000

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.

21.5.1.22 USB2_CONTROLLER_USB2D_TXFILLTUNING_0

USB2D Transmit fill tuning register

Offset: 154h | Read/Write: R/W | Reset: 0b000010xxx0000000000000

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

Bit	Reset	Description
		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. 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 1.267us when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode

21.5.1.23 USB2_CONTROLLER_USB2D_ICUSB_CTRL_0

This register enables and controls the ICUSB FS/LS transceiver.

USB2D ICUSB control register

Offset: 15ch | Read/Write: R/W | Reset: 0b0000

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.

21.5.1.24 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: 160h | 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 no. 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.

21.5.1.25 USB2_CONTROLLER_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 174h | 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.
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 didn't 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 are 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 = DISBLE 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 behaviour. 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 = DISBLE 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 behaviour. 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 = DISBLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any other value 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

Bit	R/W	Reset	Description
			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 are: 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
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

Bit	R/W	Reset	Description
			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
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

21.5.1.26 USB2_CONTROLLER_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 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 in host mode. This register shares the same address with the DEVLC register (which is used only in device mode).

Offset: 1b4h | Read/Write: R/W | Reset: 0b0000xxxx000000000000xxxxxxxxxxx0

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
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 a 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 is reserved. Value Meaning 00b Disables auto LPM. 01b If there is no activity for a certain number of SOFs the controller will send a 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).

Bit	R/W	Reset	Description
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
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, a LPM token will be sent and the port will enter in suspend state. The SOF counter for this threshold is incremented each 125us, 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 follows: 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 follows: 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

21.5.1.27 USB2_CONTROLLER_USB2D_OTGSC_0

USB2D On-The-Go (OTG) Status and Control Register

Offset: 1f4h | Read/Write: R/W | Reset: 0b0000000x0000000xxxxxxxxxx000x00

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

Bit	R/W	Reset	Description
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
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

Bit	R/W	Reset	Description
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

21.5.1.28 USB2_CONTROLLER_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 1f8h | Read/Write: R/W | Reset: 0b0xxxxxxxxx0xxxxx

Bit	R/W	Reset	Description
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 implemented for this, so SW should not use this.
4	RO	X	SDIS: Stream disbable: 1 Streaming is disabled - helpful to avoid overrun/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

Bit	R/W	Reset	Description
			1 = RESERVED 2 = DEVICE_MODE 3 = HOST_MODE

21.5.1.29 USB2_CONTROLLER_USB2D_ENDPTNAK_0

USB2D Endpoint NAK register

Offset: 200h | 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

21.5.1.30 USB2_CONTROLLER_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable register

Offset: 204h | 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

21.5.1.31 USB2_CONTROLLER_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 208h | Read/Write: R/W | Reset: 0b0000000000000000

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

Bit	Reset	Description
		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 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

Bit	Reset	Description
		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
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

21.5.1.32 USB2_CONTROLLER_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 20ch | 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.

Bit	Reset	Description
		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
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
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.

Bit	Reset	Description
		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
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

Bit	Reset	Description
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

21.5.1.33 USB2_CONTROLLER_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 210h | 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.

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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

21.5.1.34 USB2_CONTROLLER_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 214h | 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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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
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

Bit	Reset	Description
		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

21.5.1.35 USB2_CONTROLLER_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 218h | 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
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)

Bit	Reset	Description
		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
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

Bit	Reset	Description
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
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

Bit	Reset	Description
		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

21.5.1.36 USB2_CONTROLLER_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 21ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

Bit	Reset	Description
		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

21.5.1.37 USB2_CONTROLLER_USB2D_ENDPTCTRL1_0

USB2D Endpoint Control 1 Register

Offset: 220h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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.

Bit	R/W	Reset	Description
			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 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

21.5.1.38 USB2_CONTROLLER_USB2D_ENDPTCTRL2_0

USB2D Endpoint Control 2 Register

Offset: 224h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			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 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

21.5.1.39 USB2_CONTROLLER_USB2D_ENDPTCTRL3_0

USB2D Endpoint Control 3 Register

Offset: 228h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.1.40 USB2_CONTROLLER_USB2D_ENDPTCTRL4_0

USB2D Endpoint Control 4 Register

Offset: 22ch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.1.41 USB2_CONTROLLER_USB2D_ENDPTCTRL5_0

USB2D Endpoint Control 5 Register

Offset: 230h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.1.42 USB2_CONTROLLER_USB2D_ENDPTCTRL6_0

USB2D Endpoint Control 6 Register

Offset: 234h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.1.43 USB2_CONTROLLER_USB2D_ENDPTCTRL7_0

USB2D Endpoint Control 7 Register

Offset: 238h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.1.44 USB2_CONTROLLER_USB2D_ENDPTCTRL8_0

USB2D Endpoint Control 8 Register

Offset: 23ch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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.

Bit	R/W	Reset	Description
			0 = EP_OK 1 = EP_STALL

21.5.1.45 USB2_CONTROLLER_USB2D_ENDPTCTRL9_0

USB2D Endpoint Control 9 Register

Offset: 240h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 is fixed to 0.

Bit	R/W	Reset	Description
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

21.5.1.46 USB2_CONTROLLER_USB2D_ENDPTCTRL10_0

USB2D Endpoint Control 10 Register

Offset: 244h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This 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

21.5.1.47 USB2_CONTROLLER_USB2D_ENDPTCTRL11_0

USB2D Endpoint Control 11 Register

Offset: 248h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			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 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

21.5.1.48 USB2_CONTROLLER_USB2D_ENDPTCTRL12_0

USB2D Endpoint Control 12 Register

Offset: 24ch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			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 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

21.5.1.49 USB2_CONTROLLER_USB2D_ENDPTCTRL13_0

USB2D Endpoint Control 13 Register

Offset: 250h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.1.50 USB2_CONTROLLER_USB2D_ENDPTCTRL14_0

USB2D Endpoint Control 14 Register

Offset: 254h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			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 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

21.5.1.51 USB2_CONTROLLER_USB2D_ENDPTCTRL15_0

USB2D Endpoint Control 15 Register

Offset: 258h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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

Bit	R/W	Reset	Description
			3 = INTR
17	RO	X	TXD: This 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 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

21.5.2 USB1 Controller Interface Registers

These are used to generate the actual RTL registers for USB1 controller interface.

21.5.2.1 USB1_IF_USB_SUSP_CTRL_0

This register controls the suspend and resume behavior of USB controller/PHY.

USB suspend control register

Offset: 400h | Read/Write: R/W | Reset: 0b00xxxxxx000xxx01000xx000000

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

Bit	R/W	Reset	Description
			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 USB PHY clock is waked 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 wakeup 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 wakeup 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

Bit	R/W	Reset	Description
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable. If this bit is enabled, USB will wakeup 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

21.5.2.2 USB1_IF_USB_PHY_VBUS_SENSORS_0

This register controls the OTG VBUS sensors in the USB PHY. We have the following 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 _STS. The field _CHG_DET is set to 1 whenever a change is detected in the value of . 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 , it can set _SW_EN to 1, and set _SW_VALUE to 1 or 0 as per the requirement.

We have 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 approrpriate 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.

Note: The source for A_SESS_VLD sensor can be programmed according to the setting of

USB1_VBUS_SENSE_CTL in register USB1_LEGACY_CTRL.

USB PHY VBUS SENSORS control register

Offset: 404h | Read/Write: R/W | Reset: 0b0000x00x0000x00x0000x00x0000x00

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 wakeup from suspend whenever a change is detected on A_VBUS_VLD. 0 = DISABLE 1 = ENABLE

Bit	R/W	Reset	Description
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 wakeup 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.

Bit	R/W	Reset	Description
			0 = DISABLE 1 = ENABLE
14	RW	0x0	B_SESS_VLD_WAKEUP_EN: B_SESS_VLD wakeup enable. If this bit is enabled, USB will wakeup 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 wakeup 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: 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

Bit	R/W	Reset	Description
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

21.5.2.3 USB1_IF_USB_PHY_VBUS_WAKEUP_ID_0

This register controls the battery charger (VDCD_DET, VDAT_DET), VBUS_WAKEUP and ID sensors. We have the following sensors in this register:

- VBUS_WAKEUP
- ID
- VDAT_DET
- VDCD_DET

The debounced status of each sensor can be read from _STS. The field _CHG_DET is set to 1 whenever a change is detected in the value of. 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 , it can set _SW_EN to 1, and set _SW_VALUE to 1 or 0 as per the requirement.

We have 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 approrpriate 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 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: 408h | Read/Write: R/W | Reset: 0b0000x00xx000x00xx000x00x1000x00

Bit	R/W	Reset	Description
30	RW	0x0	VBUS_WAKEUP_WAKEUP_EN: VBUS_WAKEUP wakeup enable. If this bit is enabled, USB will wakeup 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

Bit	R/W	Reset	Description
			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 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

Bit	R/W	Reset	Description
			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
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

21.5.2.4 USB1_IF_USB_PHY_ALT_VBUS_STS_0

USB PHY Alternate VBUS status register

Offset: 40ch | Read/Write: RO | Reset: 0bxxxxxxxxx

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

21.5.2.5 USB1_IF_USB1_LEGACY_CTRL_0

This register controls legacy mode access for USB controller.

USB1 Legacy control register

Offset: 410h | Read/Write: R/W | Reset: 0b00x

Bit	R/W	Reset	Description
2:1	RW	0x0	USB1_VBUS_SENSE_CTL: Vbus_sense control. Controls which VBUS sensor input is driven to the controller. 00: Use VBUS_WAKEUP. 01: Use (A_SESS_VLD || B_SESS_VLD) output from the PHY if the PHY clock is available. Otherwise, use VBUS_WAKEUP. 10: Use (A_SESS_VLD || B_SESS_VLD) output from the PHY 11: Use A_SESS_VLD output from the PHY 0 = VBUS_WAKEUP 1 = AB_SESS_VLD_OR_VBUS_WAKEUP 2 = AB_SESS_VLD 3 = A_SESS_VLD
0	RO	X	USB1_NO_LEGACY_MODE: Legacy registers select. The default is to select legacy mode registers in APB_MISC for USB1. Selects new registers if this is set to 1. 0 = LEGACY 1 = NEW

21.5.2.6 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: 420h | Read/Write: R/W | Reset: 0b010010

Bit	Reset	Description
5:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. Software should not change this.

21.5.2.7 USB1_IF_USB_RSM_DLY_0

USB Controller Resume Signalling Delay

Offset: 490h | Read/Write: R/W | Reset: 0b0110100101111000

Bit	Reset	Description
15:0	0x6978	TIME_TO_RESUME: Send the resume back in no. of 60 MHz cycles. Default gives 900 usec delay. Only applicable in host mode.

21.5.2.8 USB1_IF_SPARE_0

For ICUSB PADCTLS. Spare Register

Offset: 498h | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:16	0xffff	SPARE_HI: Spare register bits, defaulted to 1
15:0	0x0	SPARE_LO: Spare register bits, defaulted to 0. 0 SPARE_LO[4] - usb_hs_rsm_eop_fix_en, Enable HS_RESUME EOP fix.

21.5.2.9 USB1_UTMIP_PLL_CFG0_0

USB_PHY PLL Configuration Register 0

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

21.5.3 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

21.5.3.1 UTMIP REGISTER: PLL_CFG0

This register was used to configure PLL inside UTMIP block prior to Tegra 3. 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.

Note: This register has been moved to the Clock and Reset section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT WHATSOEVER !

Offset: 800h | Read/Write: R/W | Reset: 0b00000000010100000000000110000000

Bit	Reset	Description
30:28	0x0	UTMIP_PLL_SELECT: Selects which of the eight 60MHz clock phases to produce at the output of the USB PHY PLL. This is for a test mode. See cell specification.

Bit	Reset	Description
27	0x0	UTMIP_PLL_PDIVRST: PDIVRST of the UTMIP PHY PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.
26:24	0x0	UTMIP_PLL_PDIV: PDIV[2:0] input of the USB_PHY PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.
23:16	0x28	UTMIP_PLL_NDIV: NDIV[7:0] input of USB_PHY 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 USB_PHY PLL. This is the predivide on the PLL. 0x0 is not allowed. See cell specification.
7	0x1	UTMIP_PLL_VCOMULTBY2: VCOMULTBY2 control of the UTMIP PHY PLL. Recommended setting is on. Additional divide by 2 on the VCO feedback. Which is setting the bit to 1. See cell spec.
6:1	0x0	UTMIP_PLL_LOCKSEL: LOCKSEL[5:0] input of USB_PHY PLL. Used in test modes only. See cell specification.
0	0x0	UTMIP_PLL_LOCKENABLE: LOCK_ENABLE input of USB_PHY PLL. Normally only used during test. See cell specification.

21.5.3.2 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 WHATSOEVER !

This register was used to configure PLL inside UTMIP block prior to Tegra 3. This has been de-featured from UTMIP space and moved to CAR space.

PLL CONFIGURATION & PARAMETERS

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

- Xtal clock -> enters PLL_U to generate 12MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock

The following parameters control the bringup of the plls:

Coming out of reset or suspend

- -> PllUOnState: start pllu_enable_count and pll_lock_count
- -> Wait ~1us to actually enable the PLL_U (pllu_enable_count == ClkXtal * PLLU_ENABLE_DLY_COUNT * 8)
- -> Wait ~1ms until PLL_U is actually stable (pll_lock_count == ClkXtal * PLLU_STABLE_COUNT * 256) => USB_PHY PLL_ENABLE pll_active_count = 0
- -> Wait 5us to enable pll_active: pll_active_count == ClkXtal * PLL_ACTIVE_DLY_COUNT * 16 => USB_PHY PLL_ACTIVE
- -> Wait ~2.5ms until USB_PHY is actually stable (pll_lock_count == ClkXtal * XTAL_FREQ_COUNT * 256) => USB_PHY

Numbers for a 19.2MHz Xtal 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
- PLL_ACTIVE_DLY_COUNT[4:0] = 5 * 19.2 / 16 = 6 = 0x06

- XTAL_FREQ_COUNT[11:0] = 2500 * 19.2 / 256 = 187 = 0xbb

Offset: 804h | Read/Write: R/W | Reset: 0b00011000001000000000000011000000

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	0x8	UTMIP_PLL_SETUP: SETUP[8:0] input of USB_PHY PLL.
17	0x0	UTMIP_FORCE_PLLU_POWERUP: Force PLL_U into power up.
16	0x0	UTMIP_FORCE_PLLU_POWERDOWN: Force PLL_U into power down. (Overrides FORCE_PLLU_POWERUP.)
15	0x0	UTMIP_FORCE_PLL_ENABLE_POWERUP: Force USB_PHY PLL pll_enable input on.
14	0x0	UTMIP_FORCE_PLL_ENABLE_POWERDOWN: Force USB_PHY PLL pll_enable input off. (Overrides FORCE_PLL_ENABLE_POWERUP.)
13	0x0	UTMIP_FORCE_PLL_ACTIVE_POWERUP: Force USB_PHY PLL pll_active input on.
12	0x0	UTMIP_FORCE_PLL_ACTIVE_POWERDOWN: Force USB_PHY 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 USB_PHY PLL is considered stable.

21.5.3.3 USB1_UTMIP_XCVR_CFG0_0

UTMIP transceiver cell configuration register 0

Offset: 808h | 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.)
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.

Bit	Reset	Description
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.

21.5.3.4 USB1_UTMIP_BIAS_CFG0_0

UTMIP Bias cell configuration register 0

Offset: 80ch | Read/Write: R/W | Reset: 0b01100000000000110000000000

Bit	Reset	Description
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.
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.

21.5.3.5 USB1_UTMIP_HSRX_CFG0_0

UTMIP High speed receive config 0

Offset: 810h | 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

Bit	Reset	Description
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 cycles of idle 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

21.5.3.6 USB1_UTMIP_HSRX_CFG1_0

UTMIP High speed receive config 1

Offset: 814h | Read/Write: R/W | Reset: 0b010011

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

21.5.3.7 USB1_UTMIP_FSLSRX_CFG0_0

UTMIP full and Low speed receive config 0

Offset: 818h | Read/Write: R/W | Reset: 0b11111101010101001000010000101001

Bit	Reset	Description
31	0x1	UTMIP_FSLS_SE1_DRIBBLE_FILTER: One SE1, don't allow dribble
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

Bit	Reset	Description
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

21.5.3.8 USB1_UTMIP_FSLSRX_CFG1_0

UTMIP full and Low speed receive config 1

Offset: 81ch | Read/Write: R/W | Reset: 0b01000100110011101010000000000

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

21.5.3.9 USB1_UTMIP_TX_CFG0_0

UTMIP transmit config signals

Offset: 820h | Read/Write: R/W | Reset: 0b00010000001000000000

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
8	0x0	UTMIP_HS_DISCON_DISABLE: Disable high speed disconnect
7	0x0	UTMIP_HS_POSTAMBLE_OUTPUT_ENABLE: output enable turns off 1 cycle after

Bit	Reset	Description
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 sent SYNC or EOP

21.5.3.10 USB1_UTMIP_MISC_CFG0_0

UTMIP miscellaneous configurations

Offset: 824h | Read/Write: R/W | Reset: 0b00000111110000000000000001111000

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: Don't block changes for 4ms 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.
8	0x0	UTMIP_FORCE_PULLDN_DM: Force DM pulldown active.

Bit	Reset	Description
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: Don't 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

21.5.3.11 USB1_UTMIP_MISC_CFG1_0

UTMIP miscellaneous configurations

Offset: 828h | Read/Write: R/W | Reset: 0b10000000001100110000000000100100

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 neg edge sync for linestate
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

21.5.3.12 USB1_UTMIP_DEBOUNCE_CFG0_0

UTMIP Avalid and Bvalid debounce

Debounce values IdDig, Avalid, Bvalid, VbusValid, VbusWakeUp, and SessEnd. Each of these signals have their 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, we have:

- BIAS_DEBOUNCE_A[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 82ch | 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

21.5.3.13 USB1_UTMIP_BAT_CHRG_CFG0_0

UTMIP battery charger configuration

Offset: 830h | Read/Write: R/W | Reset: 0b000000xx000001

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

21.5.3.14 USB1_UTMIP_SPARE_CFG0_0

UTMIP spare configuration bits

Offset: 834h | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:0	-65536	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 31 to 5: Reserved Note: 4:3 stand for using fused termination and setup value.

21.5.3.15 USB1_UTMIP_XCVR_CFG1_0

UTMIP transceiver cell configuration register 1

Offset: 838h | Read/Write: R/W | Reset: 0b00011000100000010000010101

Bit	Reset	Description
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 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.)

21.5.3.16 USB1_UTMIP_BIAS_CFG1_0

UTMIP Bias cell configuration register 1

Offset: 83ch | Read/Write: R/W | Reset: 0b00000000101101

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 20us. For a Xtal clock of 13MHz it should be set a 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.)

21.5.3.17 USB1_UTMIP_BIAS_STS0_0

UTMIP Bias cell status register 0

Offset: 840h | 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.

21.5.3.18 USB1_UTMIP_CHRG_DEB_CFG0_0

UTMIP VDcd_Det and VDat_Det debounce

Debounce values VDcd_Det and VDat_Det. Each of these signals have their own debouncer and 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: 844h | 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

21.5.3.19 USB1_UTMIP_MISC_STS0_0

UTMIP MISC status register

Offset: 848h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxx

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

21.5.3.20 USB1_UTMIP_PMC_WAKEUP0_0

UTMIP PMC Wakeup value

Offset: 84ch | Read/Write: R/W | Reset: 0bx0

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

21.5.3.21 USB2_QH_USB2D_QH_EP_0_OUT_0

USB2D Queue Head for OUT endpoint 0

Offset: 1000h | 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.

21.5.3.22 USB2_QH_USB2D_QH_EP_0_IN_0

USB2D Queue Head for IN endpoint 0

Offset: 1040h | 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.

21.5.3.23 USB2_QH_USB2D_QH_EP_1_OUT_0

USB2D Queue Head for OUT endpoint 1

Offset: 1080h | 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.

21.5.3.24 USB2_QH_USB2D_QH_EP_1_IN_0

USB2D Queue Head for IN endpoint 1

Offset: 10c0h | 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.

21.5.3.25 USB2_QH_USB2D_QH_EP_2_OUT_0

USB2D Queue Head for OUT endpoint 2

Offset: 1100h | 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.

21.5.3.26 USB2_QH_USB2D_QH_EP_2_IN_0

USB2D Queue Head for IN endpoint 2

Offset: 1140h | 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.

21.5.3.27 USB2_QH_USB2D_QH_EP_3_OUT_0

USB2D Queue Head for OUT endpoint 3

Offset: 1180h | 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.

21.5.3.28 USB2_QH_USB2D_QH_EP_3_IN_0

USB2D Queue Head for IN endpoint 3

Offset: 11c0h | 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.

21.5.3.29 USB2_QH_USB2D_QH_EP_4_OUT_0

USB2D Queue Head for OUT endpoint 4

Offset: 1200h | 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.

21.5.3.30 USB2_QH_USB2D_QH_EP_4_IN_0

USB2D Queue Head for IN endpoint 4

Offset: 1240h | 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.

21.5.3.31 USB2_QH_USB2D_QH_EP_5_OUT_0

USB2D Queue Head for OUT endpoint 5

Offset: 1280h | 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.

21.5.3.32 USB2_QH_USB2D_QH_EP_5_IN_0

USB2D Queue Head for IN endpoint 5

Offset: 12c0h | 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.

21.5.3.33 USB2_QH_USB2D_QH_EP_6_OUT_0

USB2D Queue Head for OUT endpoint 6

Offset: 1300h | 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.

21.5.3.34 USB2_QH_USB2D_QH_EP_6_IN_0

USB2D Queue Head for IN endpoint 6

Offset: 1340h | 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.

21.5.3.35 USB2_QH_USB2D_QH_EP_7_OUT_0

USB2D Queue Head for OUT endpoint 7

Offset: 1380h | 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.

21.5.3.36 USB2_QH_USB2D_QH_EP_7_IN_0

USB2D Queue Head for IN endpoint 7

Offset: 13c0h | 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.

21.5.3.37 USB2_QH_USB2D_QH_EP_8_OUT_0

USB2D Queue Head for OUT endpoint 8

Offset: 1400h | 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.

21.5.3.38 USB2_QH_USB2D_QH_EP_8_IN_0

USB2D Queue Head for IN endpoint 8

Offset: 1440h | 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.

21.5.3.39 USB2_QH_USB2D_QH_EP_9_OUT_0

USB2D Queue Head for OUT endpoint 9

Offset: 1480h | 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.

21.5.3.40 USB2_QH_USB2D_QH_EP_9_IN_0

USB2D Queue Head for IN endpoint 9

Offset: 14c0h | 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.

21.5.3.41 USB2_QH_USB2D_QH_EP_10_OUT_0

USB2D Queue Head for OUT endpoint 10

Offset: 1500h | 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.

21.5.3.42 USB2_QH_USB2D_QH_EP_10_IN_0

USB2D Queue Head for IN endpoint 10

Offset: 1540h | 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.

21.5.3.43 USB2_QH_USB2D_QH_EP_11_OUT_0

USB2D Queue Head for OUT endpoint 11

Offset: 1580h | 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.

21.5.3.44 USB2_QH_USB2D_QH_EP_11_IN_0

USB2D Queue Head for IN endpoint 11

Offset: 15c0h | 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.

21.5.3.45 USB2_QH_USB2D_QH_EP_12_OUT_0

USB2D Queue Head for OUT endpoint 12

Offset: 1600h | 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.

21.5.3.46 USB2_QH_USB2D_QH_EP_12_IN_0

USB2D Queue Head for IN endpoint 12

Offset: 1640h | 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.

21.5.3.47 USB2_QH_USB2D_QH_EP_13_OUT_0

USB2D Queue Head for OUT endpoint 13

Offset: 1680h | 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.

21.5.3.48 USB2_QH_USB2D_QH_EP_13_IN_0

USB2D Queue Head for IN endpoint 13

Offset: 16c0h | 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.

21.5.3.49 USB2_QH_USB2D_QH_EP_14_OUT_0

USB2D Queue Head for OUT endpoint 14

Offset: 1700h | 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.

21.5.3.50 USB2_QH_USB2D_QH_EP_14_IN_0

USB2D Queue Head for IN endpoint 14

Offset: 1740h | 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.

21.5.3.51 USB2_QH_USB2D_QH_EP_15_OUT_0

USB2D Queue Head for OUT endpoint 15

Offset: 1780h | 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.

21.5.3.52 USB2_QH_USB2D_QH_EP_15_IN_0

USB2D Queue Head for IN endpoint 15

Offset: 17c0h | 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.

21.5.4 USB2 Controller Registers

21.5.4.1 USB2_CONTROLLER_1_USB2D_ID_0

USB2D Identification Register

Offset: 000h | 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: Ones complement version of ID. This field is set to 0xF9.
7:0	X	ID: Configuration number. This field is set to 0x06

21.5.4.2 USB2_CONTROLLER_1_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 008h | Read/Write: RO | Reset: 0bxxxx

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.

21.5.4.3 USB2_CONTROLLER_1_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

Offset: 00ch |Read/Write: RO | Reset: 0bxxxxxx

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: No. 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.

21.5.4.4 USB2_CONTROLLER_1_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 010h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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.

21.5.4.5 USB2_CONTROLLER_1_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 014h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

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.

21.5.4.6 USB2_CONTROLLER_1_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

Offset: 100h | Read/Write: RO | Reset: 0bxxxxxxxx

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.

21.5.4.7 USB2_CONTROLLER_1_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 102h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

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.

21.5.4.8 USB2_CONTROLLER_1_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

Offset: 104h | 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.

21.5.4.9 USB2_CONTROLLER_1_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

Offset: 108h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxx

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 with 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 with 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.

Bit	Reset	Description
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".

21.5.4.10 USB2_CONTROLLER_1_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

Offset: 120h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

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.

21.5.4.11 USB2_CONTROLLER_1_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 124h | Read/Write: RO | Reset: 0bxxxxxxxxx

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 with 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.

21.5.4.12 USB2_CONTROLLER_1_USB2D_EXTSTS_0

USB2D EXTSTS Register

Offset: 128h | Read/Write: R/W | Reset: 0b0000x

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

Bit	R/W	Reset	Description
			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 readonly. 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

21.5.4.13 USB2_CONTROLLER_1_USB2D_USBEXTINTR_0

USB2D EXTINTR Register

Offset: 12ch | Read/Write: R/W | Reset: 0b00000

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
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

21.5.4.14 USB2_CONTROLLER_1_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 130h | Read/Write: R/W | Reset: 0b000000001000000x1x11x0000000

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 BA field of 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 a 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

Bit	R/W	Reset	Description
			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 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

Bit	R/W	Reset	Description
			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) 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

21.5.4.15 USB2_CONTROLLER_1_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 134h | Read/Write: R/W | Reset: 0b00000000000000000x1x0x0000x0000

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.

Bit	R/W	Reset	Description
			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
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 a 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 125µs 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 125µs 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

21.5.4.16 USB2_CONTROLLER_1_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 138h | Read/Write: R/W | Reset: 0b0000000000000000xxxxx0x000000000

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.
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 SW 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 SW 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

Bit	Reset	Description
		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

21.5.4.17 USB2_CONTROLLER_1_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 13ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

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.

21.5.4.18 USB2_CONTROLLER_1_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 144h | Read/Write: R/W | Reset: 0b00000000000000000000

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

Bit	Reset	Description
		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 can not 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.

21.5.4.19 USB2_CONTROLLER_1_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 148h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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.

21.5.4.20 USB2_CONTROLLER_1_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 14ch | Read/Write: R/W | Reset: 0b0x

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.

21.5.4.21 USB2_CONTROLLER_1_USB2D_BURSTSIZE_0

USB2D Burst Size register

Offset: 150h | Read/Write: R/W | Reset: 0b0000100000001000

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.

21.5.4.22 USB2_CONTROLLER_1_USB2D_TXFILLTUNING_0

USB2D Transmit fill tuning register

Offset: 154h | Read/Write: R/W | Reset: 0b000010xxx0000000000000

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 unpredicable 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. 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 1.267us when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode

21.5.4.23 USB2_CONTROLLER_1_USB2D_ICUSB_CTRL_0

USB2D ICUSB control register

This register enables and controls the ICUSB FS/LS transceiver.

Offset: 15ch | Read/Write: R/W | Reset: 0b0000

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.

21.5.4.24 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 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: 160h | 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 can not 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 can not 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 no. This field should be always written as 0.

Bit	R/W	Reset	Description
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.

21.5.4.25 USB2_CONTROLLER_1_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 174h | 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.
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 didn't 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 are 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 = DISBLE 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 behaviour. 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 = DISBLE 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 behaviour. 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 = DISBLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any other value 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

Bit	R/W	Reset	Description
			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). 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 are: 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
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

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
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

Bit	R/W	Reset	Description
			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

21.5.4.26 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: 1b4h | Read/Write: R/W | Reset: 0b0000xxxx000000000000xxxxxxxxxxx0

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
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 a 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

Bit	R/W	Reset	Description
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 a 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
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, a LPM token will be sent and the port will enter in suspend state. The SOF counter for this threshold is incremented each 125us, 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 follows: 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 follows: 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

21.5.4.27 USB2_CONTROLLER_1_USB2D_OTGSC_0

USB2D On-The-Go (OTG) Status and Control Register

Offset: 1f4h | Read/Write: R/W | Reset: 0b0000000x0000000xxxxxxxxxx000x00

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

Bit	R/W	Reset	Description
			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
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

Bit	R/W	Reset	Description
			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

21.5.4.28 USB2_CONTROLLER_1_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 1f8h | Read/Write: R/W | Reset: 0b0xxxxxxxxx0xxxxx

Bit	R/W	Reset	Description
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 implemented for this, so SW should not use this.
4	RO	X	SDIS: Stream disbable: 1 Streaming is disabled - helpful to avoid overrun/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

Bit	R/W	Reset	Description
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

21.5.4.29 USB2_CONTROLLER_1_USB2D_ENDPTNAK_0

USB2D Endpoint NAK register

Offset: 200h | 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

21.5.4.30 USB2_CONTROLLER_1_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable register

Offset: 204h | 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

21.5.4.31 USB2_CONTROLLER_1_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 208h | Read/Write: R/W | Reset: 0b0000000000000000

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

Bit	Reset	Description
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 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

Bit	Reset	Description
		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
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

21.5.4.32 USB2_CONTROLLER_1_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 20ch | 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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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

21.5.4.33 USB2_CONTROLLER_1_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 210h | 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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
0	0x0	FERB0: Flush Endpoint Receive Buffer: Writing a one to this bit causes the associated endpoint to clear

Bit	Reset	Description
		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

21.5.4.34 USB2_CONTROLLER_1_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 214h | 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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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
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

Bit	Reset	Description
		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

21.5.4.35 USB2_CONTROLLER_1_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 218h | 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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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

21.5.4.36 USB2_CONTROLLER_1_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 21ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

21.5.4.37 USB2_CONTROLLER_1_USB2D_ENDPTCTRL1_0

USB2D Endpoint Control 1 Register

Offset: 220h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			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 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

21.5.4.38 USB2_CONTROLLER_1_USB2D_ENDPTCTRL2_0

USB2D Endpoint Control 2 Register

Offset: 224h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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

Bit	R/W	Reset	Description
			3 = INTR
17	RO	X	TXD: This 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 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

21.5.4.39 USB2_CONTROLLER_1_USB2D_ENDPTCTRL3_0

USB2D Endpoint Control 3 Register

Offset: 228h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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

Bit	R/W	Reset	Description
			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 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 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

21.5.4.40 USB2_CONTROLLER_1_USB2D_ENDPTCTRL4_0

USB2D Endpoint Control 4 Register

Offset: 22ch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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

Bit	R/W	Reset	Description
			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 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 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

21.5.4.41 USB2_CONTROLLER_1_USB2D_ENDPTCTRL5_0

USB2D Endpoint Control 5 Register

Offset: 230h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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

Bit	R/W	Reset	Description
			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 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 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

21.5.4.42 USB2_CONTROLLER_1_USB2D_ENDPTCTRL6_0

USB2D Endpoint Control 6 Register

Offset: 234h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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.

Bit	R/W	Reset	Description
			0 = EP_OK 1 = EP_STALL

21.5.4.43 USB2_CONTROLLER_1_USB2D_ENDPTCTRL7_0

USB2D Endpoint Control 7 Register

Offset: 238h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 is fixed to 0.

Bit	R/W	Reset	Description
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

21.5.4.44 USB2_CONTROLLER_1_USB2D_ENDPTCTRL8_0

USB2D Endpoint Control 8 Register

Offset: 23ch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This 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

21.5.4.45 USB2_CONTROLLER_1_USB2D_ENDPTCTRL9_0

USB2D Endpoint Control 9 Register

Offset: 240h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			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 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

21.5.4.46 USB2_CONTROLLER_1_USB2D_ENDPTCTRL10_0

USB2D Endpoint Control 10 Register

Offset: 244h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			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 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

21.5.4.47 USB2_CONTROLLER_1_USB2D_ENDPTCTRL11_0

USB2D Endpoint Control 11 Register

Offset: 248h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.4.48 USB2_CONTROLLER_1_USB2D_ENDPTCTRL12_0

USB2D Endpoint Control 12 Register

Offset: 24ch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			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 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

21.5.4.49 USB2_CONTROLLER_1_USB2D_ENDPTCTRL13_0

USB2D Endpoint Control 13 Register

Offset: 250h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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

Bit	R/W	Reset	Description
			3 = INTR
17	RO	X	TXD: This 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 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

21.5.4.50 USB2_CONTROLLER_1_USB2D_ENDPTCTRL14_0

USB2D Endpoint Control 14 Register

Offset: 254h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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

Bit	R/W	Reset	Description
			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 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 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

21.5.4.51 USB2_CONTROLLER_1_USB2D_ENDPTCTRL15_0

USB2D Endpoint Control 15 Register

Offset: 258h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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

Bit	R/W	Reset	Description
			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 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 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

21.5.5 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 whatsoever to assert ULPIS2S_LINE_RESET.

21.5.5.1 USB2_IF_USB_SUSP_CTRL_0

This register controls the suspend and reusume behavior of USB controller/PHY.

USB suspend control register

Offset: 400h | Read/Write: R/W | Reset: 0b00111110000x1001000xx000000

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's 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
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

Bit	R/W	Reset	Description
			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 waked 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 wakeup 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 wakeup 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

Bit	R/W	Reset	Description
2	RW	0x0	USB_WAKE_ON_RESUME_EN: Wake on resume enable. If this bit is enabled, USB will wakeup 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

21.5.5.2 USB2_IF_USB_PHY_VBUS_SENSORS_0

USB PHY VBUS SENSORS control register

This register controls the OTG VBUS sensors in the USB PHY. We have the following 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 _STS. The field _CHG_DET is set to 1 whenever a change is detected in the value of. 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 , it can set _SW_EN to 1, and set _SW_VALUE to 1 or 0 as per the requirement.

We have 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 apprORpriate 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.

Note: The source for A_SESS_VLD sensor can be programmed according to the setting of USB1_VBUS_SENSE_CTL in register USB1_LEGACY_CTRL.

Offset: 404h | Read/Write: R/W | Reset: 0b0000x00x0000x00x0000x00x0000x00

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 wakeup 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

Bit	R/W	Reset	Description
			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 wakeup 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

Bit	R/W	Reset	Description
			1 = ENABLE
14	RW	0x0	B_SESS_VLD_WAKEUP_EN: B_SESS_VLD wakeup enable. If this bit is enabled, USB will wakeup 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 wakeup 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: 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

Bit	R/W	Reset	Description
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

21.5.5.3 USB2_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. We have the following sensors in this register:

- VBUS_WAKEUP
- ID
- VDAT_DET
- VDCD_DET

The debounced status of each sensor can be read from _STS. The field _CHG_DET is set to 1 whenever a change is detected in the value of. 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 , it can set _SW_EN to 1, and set _SW_VALUE to 1 or 0 as per the requirement.

We have 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 approrpriate 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 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: 408h | Read/Write: R/W | Reset: 0b0000x00xx000x00xx000x00x1000x00

Bit	R/W	Reset	Description
30	RW	0x0	VBUS_WAKEUP_WAKEUP_EN: VBUS_WAKEUP wakeup enable. If this bit is enabled, USB will wakeup 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

Bit	R/W	Reset	Description
			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 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

Bit	R/W	Reset	Description
			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
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

21.5.5.4 USB2_IF_USB_PHY_ALT_VBUS_STS_0

USB PHY Alternate VBUS status register

Offset: 40ch | Read/Write: RO | Reset: 0bxxxxxxxxx

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

21.5.5.5 USB2_IF_USB_ULPIS2S_CTRL_0

This register is used to setup 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: 418h | Read/Write: R/W | Reset: 0b0000xx0000000000xxxx0000

Bit	Reset	Description
23:20	0x0	ULPIS2S_CLAMP_LINE_DRIVE: The line drive value that should be sent into the line simulator during xmit clamping. The suggested value is 0x1: tri-state.
17	0x0	ULPIS2S_SLV1_CLAMP_XMIT: When set to 1, the outputs of the SLV1 xmit statemachine 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 xmit statemachine 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.

Bit	Reset	Description
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's supported in Opmode3. All other Opmode3 generate packets are not supported under any circumstances. 0 = DISABLE 1 = ENABLE
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 won't 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 bit 0: Vbus. Set to 1 to set the Vbus State field to 2'b11 in the RxCmd byte. bit 1: IdGnd. Set to 1 to set the ID field to 1'b1 in the RxCmd byte.
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's 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 enables, 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

21.5.5.6 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: 41ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	ULPIS2S_SLV1_VENDOR_ID: PHY vendor_id as seen by external ULPI master
15:0	0x0	ULPIS2S_SLV1_PRODUCT_ID: PHY product_id as seen by external ULPI master

21.5.5.7 USB2_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 as device never transmits two packets in a row.

Offset: 420h | Read/Write: R/W | Reset: 0b010010

Bit	Reset	Description
5:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. This is valid only for UHSIC PHY. Software should not change this.

21.5.5.8 USB2_IF_USB_RSM_DLY_0

USB Controller Resume Signalling Delay

Offset: 490h | Read/Write: R/W | Reset: 0b0110100101111000

Bit	Reset	Description
15:0	0x6978	TIME_TO_RESUME: Send the resume back in no. of 60 MHz cycles. Default gives 900 usec delay. Only applicable in host mode.

21.5.5.9 USB2_IF_SPARE_0

For ICUSB PADCTLS

Spare Register

Offset: 498h | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:16	0xffff	SPARE_HI: Spare register bits, defaulted to 1
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.

21.5.6 USB 2 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

21.5.6.1 USB2_UTMIP_PLL_CFG0_0

USB_PHY PLL Configuration Register 0 The data sampling frequency relies on a 960MHz clock, so the goal of the PLL is to have: In_Frequency*(PLL_VCOMULTBY2+1)*(PLL_NDIV/PLL_MDIV) = 960MHz With a 12MHz input from PLL_U, the default setting of PLL_VCOMULTBY2=1, PLL_NDIV=40 and PLL_MDIV=1 results in a correct output.

Note: This register has been moved to the CAR Section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT WHATSOEVER !!!

This register was used to configure PLL inside UTMIP block prior to Tegra 3. 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.

Offset: 800h | Read/Write: R/W | Reset: 0b00000000010100000000000110000000

Bit	Reset	Description
30:28	0x0	UTMIP_PLL_SELECT: Selects which of the eight 60MHz clock phases to produce at the output of the USB PHY PLL. This is for a test mode. See cell specification.
27	0x0	UTMIP_PLL_PDIVRST: PDIVRST of the UTMIP PHY PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.

Bit	Reset	Description
26:24	0x0	UTMIP_PLL_PDIV: PDIV[2:0] input of the USB_PHY PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.
23:16	0x28	UTMIP_PLL_NDIV: NDIV[7:0] input of USB_PHY 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 USB_PHY PLL. This is the predivide on the PLL. 0x0 is not allowed. See cell specification.
7	0x1	UTMIP_PLL_VCOMULTBY2: VCOMULTBY2 control of the UTMIP PHY PLL. Recommended setting is on. Additional divide by 2 on the VCO feedback. Which is setting the bit to 1. See cell spec.
6:1	0x0	UTMIP_PLL_LOCKSEL: LOCKSEL[5:0] input of USB_PHY PLL. Used in test modes only. See cell specification.
0	0x0	UTMIP_PLL_LOCKENABLE: LOCK_ENABLE input of USB_PHY PLL. Normally only used during test. See cell specification.

21.5.6.2 USB2_UTMIP_PLL_CFG1_0

UTMIP PLL and PLLU configuration register 1

Note: This register has been moved to the CAR Section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT WHATSOEVER !!!

This register was used to configure PLL inside UTMIP block prior to Tegra 3. This has been de-featured from UTMIP space and moved to CAR space.

PLL CONFIGURATION & PARAMETERS

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

Xtal clock -> enters PLL_U to generate 12MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock

The following parameters control the bringup of the plls:

Coming out of reset or suspend

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

pll_active_count = 0

- -> Wait 5us to enable pll_active: pll_active_count == ClkXtal * PLL_ACTIVE_DLY_COUNT * 16 => USB_PHY PLL_ACTIVE
- -> Wait ~2.5ms until USB_PHY is actually stable (pll_lock_count == ClkXtal * XTAL_FREQ_COUNT * 256) => USB_PHY

Numbers for a 19.2MHz Xtal 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)
- PLL_ACTIVE_DLY_COUNT[4:0] = 5 * 19.2 / 16 = 6 = 0x06
- XTAL_FREQ_COUNT[11:0] = 2500 * 19.2 / 256 = 187 = 0xbb

Offset: 804h | Read/Write: R/W | Reset: 0b00011000001000000000000011000000

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	0x8	UTMIP_PLL_SETUP: SETUP[8:0] input of USB_PHY PLL.
17	0x0	UTMIP_FORCE_PLLU_POWERUP: Force PLL_U into power up.
16	0x0	UTMIP_FORCE_PLLU_POWERDOWN: Force PLL_U into power down. (Overrides FORCE_PLLU_POWERUP.)
15	0x0	UTMIP_FORCE_PLL_ENABLE_POWERUP: Force USB_PHY PLL pll_enable input on.
14	0x0	UTMIP_FORCE_PLL_ENABLE_POWERDOWN: Force USB_PHY PLL pll_enable input off. (Overrides FORCE_PLL_ENABLE_POWERUP.)
13	0x0	UTMIP_FORCE_PLL_ACTIVE_POWERUP: Force USB_PHY PLL pll_active input on.
12	0x0	UTMIP_FORCE_PLL_ACTIVE_POWERDOWN: Force USB_PHY 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 USB_PHY PLL is considered stable.

21.5.6.3 USB2_UTMIP_XCVR_CFG0_0

UTMIP transceiver cell configuration register 0

Offset: 808h | 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.)
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.

Bit	Reset	Description
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.

21.5.6.4 USB2_UTMIP_BIAS_CFG0_0

UTMIP Bias cell configuration register 0

Offset: 80ch | Read/Write: R/W | Reset: 0b01100000000000110000000000

Bit	Reset	Description
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.
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.

21.5.6.5 USB2_UTMIP_HSRX_CFG0_0

UTMIP High speed receive config 0

Offset: 810h | Read/Write: R/W | Reset: 0b10010000101100101001101000000000

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

Bit	Reset	Description
20	0x0	UTMIP_NO_STRIPPING: Do not strip incoming data
19:15	0xa	UTMIP_IDLE_WAIT: Number of cycles of idle 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

21.5.6.6 USB2_UTMIP_HSRX_CFG1_0

UTMIP High speed receive config 1

Offset: 814h | Read/Write: R/W | Reset: 0b010011

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

21.5.6.7 USB2_UTMIP_FSLSRX_CFG0_0

UTMIP full and Low speed receive config 0

Offset: 818h | Read/Write: R/W | Reset: 0b11111101010101001000010000101001

Bit	Reset	Description
31	0x1	UTMIP_FSLS_SE1_DRIBBLE_FILTER: One SE1, don't allow dribble
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 of SEO should exceed the time limit

Bit	Reset	Description
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

21.5.6.8 USB2_UTMIP_FSLSRX_CFG1_0

UTMIP full and Low speed receive config 1

Offset: 81ch | Read/Write: R/W | Reset: 0b01000100110011101000000000000

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

21.5.6.9 USB2_UTMIP_TX_CFG0_0

UTMIP transmit config signals

Offset: 820h | Read/Write: R/W | Reset: 0b00010000001000000000

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
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

Bit	Reset	Description
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 sent SYNC or EOP

21.5.6.10 USB2_UTMIP_MISC_CFG0_0

UTMIP miscellaneous configurations

Offset: 824h | Read/Write: R/W | Reset: 0b00000111110000000000000001111000

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: Don't block changes for 4ms 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.
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.

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: Don't 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

21.5.6.11 USB2_UTMIP_MISC_CFG1_0

UTMIP miscellaneous configurations

Offset: 828h | Read/Write: R/W | Reset: 0b10000000001100110000000000100100

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 neg edge sync for linestate
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

21.5.6.12 USB2_UTMIP_DEBOUNCE_CFG0_0

UTMIP Avalid and Bvalid debounce

Debounce values IdDig, Avalid, Bvalid, VbusValid, VbusWakeUp, and SessEnd.

Each of these signals have their 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, we have: BIAS_DEBOUNCE_A[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 82ch | 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

21.5.6.13 USB2_UTMIP_BAT_CHRG_CFG0_0

UTMIP battery charger configuration

Offset: 830h | Read/Write: R/W | Reset: 0b000000xx000001

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

21.5.6.14 USB2_UTMIP_SPARE_CFG0_0

Utmip spare configuration bits

Offset: 834h | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:0	-65536	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 31 to 5: Reserved Note: 4:3 stand for using fused termination and setup value.

21.5.6.15 USB2_UTMIP_XCVR_CFG1_0

UTMIP transceiver cell configuration register 1

Offset: 838h | Read/Write: R/W | Reset: 0b00011000100000010000010101

Bit	Reset	Description
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 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 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.)

21.5.6.16 USB2_UTMIP_BIAS_CFG1_0

UTMIP Bias cell configuration register 1

Offset: 83ch | Read/Write: R/W | Reset: 0b00000000101101

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 20us. For a Xtal clock of 13MHz it should be set a 5.
2	0x1	UTMIP_VBUS_WAKEUP_POWERDOWN: Force VBUS_WAKEUP input into power down. 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.)

21.5.6.17 USB2_UTMIP_BIAS_STS0_0

UTMIP Bias cell status register 0

Offset: 840h | 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.

21.5.6.18 USB2_UTMIP_CHRG_DEB_CFG0_0

UTMIP VDcd_Det and VDat_Det debounce

Debounce values VDcd_Det and VDat_Det

Each of these signals have their own debouncer and 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: 844h | 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

21.5.6.19 USB2_UTMIP_MISC_STS0_0

UTMIP MISC status reg

Offset: 848h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxx

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

21.5.6.20 USB2_UTMIP_PMC_WAKEUP0_0

UTMIP PMC Wakeup value

Offset: 84ch | Read/Write: R/W | Reset: 0bx0

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

21.5.7 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

21.5.7.1 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: c00h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	UHSIC_PLL_SPARE: Reserved

21.5.7.2 USB2_UHSIC_PLL_CFG1_0

UHSIC PLL and PLLU configuration register 1

PLL CONFIGURATION & PARAMETERS

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

Xtal clock -> enters PLLU to generate 12MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock

The following parameters control the bringup of the plls:

Coming out of reset or suspend

- -> PllUOnState: start pllu_enable_count and pll_lock_count
- -> Wait ~1us 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 Xtal 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: c04h | Read/Write: R/W | Reset: 0b00011000000011000000

Bit	Reset	Description
18:14	0x3	UHSIC_PLLU_ENABLE_DLY_COUNT: 1 us / (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

21.5.7.3 USB2_UHSIC_HSRX_CFG0_0

UHSIC High speed receive config 0

Offset: c08h | Read/Write: R/W | Reset: 0b00101001110000111000

Bit	Reset	Description
18	0x0	UHSIC_NO_STRIPPING: Do not strip incoming data
17:13	0xa	UHSIC_IDLE_WAIT: Number of cycles of idle 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.

21.5.7.4 USB2_UHSIC_HSRX_CFG1_0

UHSIC High speed receive config 1

Offset: c0ch | Read/Write: R/W | Reset: 0b10001000001010010001010011

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: Nr 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 LineState 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

21.5.7.5 USB2_UHSIC_TX_CFG0_0

UHSIC transmit config signals

Offset: c10h | Read/Write: R/W | Reset: 0b1000000000

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

Bit	Reset	Description
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
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 sent SYNC or EOP

21.5.7.6 USB2_UHSIC_MISC_CFG0_0

UHSIC miscellaneous configurations

Offset: c14h | Read/Write: R/W | Reset: 0b00100001001110100011110

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 buskeeper 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 a 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 cycles of crystal clock of signal not changing to consider stable.
1	0x1	UHSIC_STABLE_ALL: Determines if all signal need to be stable to not change a config.

Bit	Reset	Description
0	0x0	UHSIC_COMB_TERMS: Use combinational terminations or synced through CLKXTAL

21.5.7.7 USB2_UHSIC_MISC_CFG1_0

UHSIC miscellaneous configurations

Offset: c18h | Read/Write: R/W | Reset: 0b100001100000000010

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: WRONG! This should be 1ms -> 0x50
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

21.5.7.8 USB2_UHSIC_PADS_CFG0_0

UHSIC Pads settings

Offset: c1ch | 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

21.5.7.9 USB2_UHSIC_PADS_CFG1_0

UHSIC Pads settings

Offset: c20h | Read/Write: R/W | Reset: 0b0011001111101

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

Bit	Reset	Description
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
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

21.5.7.10 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 Xtal clocks, enable bus keepers

Offset: c24h | Read/Write: R/W | Reset: 0b0000101

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.

21.5.7.11 USB2_UHSIC_STAT_CFG0_0

Offset: c28h | 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

21.5.7.12 USB2_UHSIC_SPARE_CFG0_0

UTMIP spare configurations, spare configuration bits

Offset: c2ch | 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 Bit 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

21.5.7.13 USB2_UHSIC_MISC_STS0_0

UHSIC SPARE Fuse value

Offset: c30h | Read/Write: RO | Reset: 0bx

Bit	Reset	Description
0	X	UHSIC_TX_HS_VLD: Indicates when the controller is driving on the bus

21.5.7.14 USB2_UHSIC_PMC_WAKEUP0_0

UHSIC PMC Wakeup value

Offset: c34h | Read/Write: R/W | Reset: 0bx0

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

21.5.7.15 USB2_QH_USB2D_QH_EP_0_OUT_0

USB2D Queue Head for OUT endpoint 0

Offset: 1000h | 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.

21.5.7.16 USB2_QH_USB2D_QH_EP_0_IN_0

USB2D Queue Head for IN endpoint 0

Offset: 1040h | 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.

21.5.7.17 USB2_QH_USB2D_QH_EP_1_OUT_0

USB2D Queue Head for OUT endpoint 1

Offset: 1080h | 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.

21.5.7.18 USB2_QH_USB2D_QH_EP_1_IN_0

USB2D Queue Head for IN endpoint 1

Offset: 10c0h | 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.

21.5.7.19 USB2_QH_USB2D_QH_EP_2_OUT_0

USB2D Queue Head for OUT endpoint 2

Offset: 1100h | 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.

21.5.7.20 USB2_QH_USB2D_QH_EP_2_IN_0

USB2D Queue Head for IN endpoint 2

Offset: 1140h | 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.

21.5.7.21 USB2_QH_USB2D_QH_EP_3_OUT_0

USB2D Queue Head for OUT endpoint 3

Offset: 1180h | 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.

21.5.7.22 USB2_QH_USB2D_QH_EP_3_IN_0

USB2D Queue Head for IN endpoint 3

Offset: 11c0h | 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.

21.5.7.23 USB2_QH_USB2D_QH_EP_4_OUT_0

USB2D Queue Head for OUT endpoint 4

Offset: 1200h | 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.

21.5.7.24 USB2_QH_USB2D_QH_EP_4_IN_0

USB2D Queue Head for IN endpoint 4

Offset: 1240h | 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.

21.5.7.25 USB2_QH_USB2D_QH_EP_5_OUT_0

USB2D Queue Head for OUT endpoint 5

Offset: 1280h | 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.

21.5.7.26 USB2_QH_USB2D_QH_EP_5_IN_0

USB2D Queue Head for IN endpoint 5

Offset: 12c0h | 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.

21.5.7.27 USB2_QH_USB2D_QH_EP_6_OUT_0

USB2D Queue Head for OUT endpoint 6

Offset: 1300h | 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.

21.5.7.28 USB2_QH_USB2D_QH_EP_6_IN_0

USB2D Queue Head for IN endpoint 6

Offset: 1340h | 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.

21.5.7.29 USB2_QH_USB2D_QH_EP_7_OUT_0

USB2D Queue Head for OUT endpoint 7

Offset: 1380h | 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.

21.5.7.30 USB2_QH_USB2D_QH_EP_7_IN_0

USB2D Queue Head for IN endpoint 7

Offset: 13c0h | 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.

21.5.7.31 USB2_QH_USB2D_QH_EP_8_OUT_0

USB2D Queue Head for OUT endpoint 8

Offset: 1400h | 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.

21.5.7.32 USB2_QH_USB2D_QH_EP_8_IN_0

USB2D Queue Head for IN endpoint 8

Offset: 1440h | 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.

21.5.7.33 USB2_QH_USB2D_QH_EP_9_OUT_0

USB2D Queue Head for OUT endpoint 9

Offset: 1480h | 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.

21.5.7.34 USB2_QH_USB2D_QH_EP_9_IN_0

USB2D Queue Head for IN endpoint 9

Offset: 14c0h | 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.

21.5.7.35 USB2_QH_USB2D_QH_EP_10_OUT_0

USB2D Queue Head for OUT endpoint 10

Offset: 1500h | 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.

21.5.7.36 USB2_QH_USB2D_QH_EP_10_IN_0

USB2D Queue Head for IN endpoint 10

Offset: 1540h | 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.

21.5.7.37 USB2_QH_USB2D_QH_EP_11_OUT_0

USB2D Queue Head for OUT endpoint 11

Offset: 1580h | 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.

21.5.7.38 USB2_QH_USB2D_QH_EP_11_IN_0

USB2D Queue Head for IN endpoint 11

Offset: 15c0h | 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.

21.5.7.39 USB2_QH_USB2D_QH_EP_12_OUT_0

USB2D Queue Head for OUT endpoint 12

Offset: 1600h | 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.

21.5.7.40 USB2_QH_USB2D_QH_EP_12_IN_0

USB2D Queue Head for IN endpoint 12

Offset: 1640h | 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.

21.5.7.41 USB2_QH_USB2D_QH_EP_13_OUT_0

USB2D Queue Head for OUT endpoint 13

Offset: 1680h | 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.

21.5.7.42 USB2_QH_USB2D_QH_EP_13_IN_0

USB2D Queue Head for IN endpoint 13

Offset: 16c0h | 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.

21.5.7.43 USB2_QH_USB2D_QH_EP_14_OUT_0

USB2D Queue Head for OUT endpoint 14

Offset: 1700h | 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.

21.5.7.44 USB2_QH_USB2D_QH_EP_14_IN_0

USB2D Queue Head for IN endpoint 14

Offset: 1740h | 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.

21.5.7.45 USB2_QH_USB2D_QH_EP_15_OUT_0

USB2D Queue Head for OUT endpoint 15

Offset: 1780h | 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.

21.5.7.46 USB2_QH_USB2D_QH_EP_15_IN_0

USB2D Queue Head for IN endpoint 15

Offset: 17c0h | 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.

21.5.8 USB3 Controller Registers

21.5.8.1 USB2_CONTROLLER_2_USB2D_ID_0

USB2D Identification Register

Offset: 000h | 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: Ones complement version of ID. This field is set to 0xF9.
7:0	X	ID: Configuration number. This field is set to 0x06

21.5.8.2 USB2_CONTROLLER_2_USB2D_HW_HOST_0

USB2D Hardware Host Register

Offset: 008h | Read/Write: RO | Reset: 0bxxxx

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.

21.5.8.3 USB2_CONTROLLER_2_USB2D_HW_DEVICE_0

USB2D Hardware Device Register

Offset: 00ch | Read/Write: RO | Reset: 0bxxxxxx

Bit	Reset	Description
5:1	X	DEVEP: VUSB_HS_DV_EP: No. 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.

21.5.8.4 USB2_CONTROLLER_2_USB2D_HW_TXBUF_0

USB2D Hardware TX Buffer Register

Offset: 010h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	TXCHANADD: VUSB_HS_TX_CHAN_ADD: Total no. 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 no. 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.

21.5.8.5 USB2_CONTROLLER_2_USB2D_HW_RXBUF_0

USB2D RX Buffer HW Parameters Register

Offset: 014h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	RXADD: VUSB_HS_RX_ADD: Total no. 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.

21.5.8.6 USB2_CONTROLLER_2_USB2D_CAPLENGTH_0

USB2D Capability Register Length Register

Offset: 100h | Read/Write: RO | Reset: 0bxxxxxxxx

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.

21.5.8.7 USB2_CONTROLLER_2_USB2D_HCIVERSON_0

USB2D Host Interface Version Number Register

Offset: 102h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

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.

21.5.8.8 USB2_CONTROLLER_2_USB2D_HCSPARAMS_0

USB2D Host Control Structural Parameters Register

Offset: 104h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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.

21.5.8.9 USB2_CONTROLLER_2_USB2D_HCCPARAMS_0

USB2D Host Control Capability Parameters Register

Offset: 108h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxx

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 with 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 with 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".

21.5.8.10 USB2_CONTROLLER_2_USB2D_DCIVERSION_0

USB2D Device Interface Version Number Register

Offset: 120h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

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.

21.5.8.11 USB2_CONTROLLER_2_USB2D_DCCPARAMS_0

USB2D Device Control Capabilities Register

Offset: 124h | Read/Write: RO | Reset: 0bxxxxxxxxx

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 with 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.

21.5.8.12 USB2_CONTROLLER_2_USB2D_EXTSTS_0

USB2D EXTSTS Register

Offset: 128h | Read/Write: R/W | Reset: 0b0000x

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 readonly. 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

21.5.8.13 USB2_CONTROLLER_2_USB2D_USBEXTINTR_0

USB2D EXTINTR Register

Offset: 12ch | Read/Write: R/W | Reset: 0b00000

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
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

21.5.8.14 USB2_CONTROLLER_2_USB2D_USBCMD_0

USB2D USB Command Register

Offset: 130h | Read/Write: R/W | Reset: 0b000000001000000x1x11x0000000

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 BA field of 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 a 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 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.

Bit	R/W	Reset	Description
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) 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

Bit	R/W	Reset	Description
			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

21.5.8.15 USB2_CONTROLLER_2_USB2D_USBSTS_0

USB2D USB Status Register

Offset: 134h | Read/Write: R/W | Reset: 0b00000000000000000x1x0x0000x0000

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 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

Bit	R/W	Reset	Description
			1 = HALTED
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 a 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 125us 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 125us 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
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

Bit	R/W	Reset	Description
			(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

21.5.8.16 USB2_CONTROLLER_2_USB2D_USBINTR_0

USB2D USB Interrupt Enable Register

Offset: 138h | Read/Write: R/W | Reset: 0b0000000000000000xxxxx0x000000000

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.
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 SW 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 SW 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

Bit	Reset	Description
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

21.5.8.17 USB2_CONTROLLER_2_USB2D_FRINDEX_0

USB2D USB Frame Index Register

Offset: 13ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

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.

21.5.8.18 USB2_CONTROLLER_2_USB2D_PERIODICLISTBASE_0

USB2D Host Controller Frame List Base Address Register

Offset: 144h | Read/Write: R/W | Reset: 0b00000000000000000000

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 can not 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.

21.5.8.19 USB2_CONTROLLER_2_USB2D_ASYNCLISTADDR_0

USB2D Next Asynchronous List Address Register

Offset: 148h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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.

21.5.8.20 USB2_CONTROLLER_2_USB2D_ASYNCTTSTS_0

USB2D Asynchronous Buffer Status for Embedded TT Register

Offset: 14ch | Read/Write: R/W | Reset: 0b0x

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.

21.5.8.21 USB2_CONTROLLER_2_USB2D_BURSTSIZE_0

USB2D Burst Size register

Offset: 150h | Read/Write: R/W | Reset: 0b0000100000001000

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.

21.5.8.22 USB2_CONTROLLER_2_USB2D_TXFILLTUNING_0

USB2D Transmit fill tuning register

Offset: 154h | Read/Write: R/W | Reset: 0b000010xxx0000000000000

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 unpredicable 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. 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 1.267us when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode

21.5.8.23 USB2_CONTROLLER_2_USB2D_ICUSB_CTRL_0

USB2D ICUSB control register

This register enables and controls the ICUSB FS/LS transceiver.

Offset: 15ch | Read/Write: R/W | Reset: 0b0000

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.

21.5.8.24 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 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.

Offset: 160h | 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 no. 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.

21.5.8.25 USB2_CONTROLLER_2_USB2D_PORTSC1_0

USB2D Port Status/Control 1 Register

Offset: 174h | 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.
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 didn't 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 are 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 = DISBLE 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 behaviour. 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 = DISBLE 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 behaviour. 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 = DISBLE 1 = ENABLE
19:16	RW	0x0	PTC: Port Test Control: Any other value 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

Bit	R/W	Reset	Description
			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 are: 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
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

Bit	R/W	Reset	Description
			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
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

21.5.8.26 USB2_CONTROLLER_2_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: 1b4h | Read/Write: R/W | Reset: 0b0000xxxx000000000000xxxxxxxxxxx0

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
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 a 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 is reserved. Value Meaning 00b Disables auto LPM. 01b If there is no activity for a certain number of SOFs the controller will send a 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).

Bit	R/W	Reset	Description
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
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, a LPM token will be sent and the port will enter in suspend state. The SOF counter for this threshold is incremented each 125us, 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 follows: 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 follows: 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

21.5.8.27 USB2_CONTROLLER_2_USB2D_OTGSC_0

USB2D On-The-Go (OTG) Status and Control Register

Offset: 1f4h | Read/Write: R/W | Reset: 0b0000000x0000000xxxxxxxxxx000x00

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

Bit	R/W	Reset	Description
			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
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

Bit	R/W	Reset	Description
			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

21.5.8.28 USB2_CONTROLLER_2_USB2D_USBMODE_0

USB2D USB Device Mode Register

Offset: 1f8h | Read/Write: R/W | Reset: 0b0xxxxxxxxx0xxxxx

Bit	R/W	Reset	Description
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 implemented for this, so SW should not use this.
4	RO	X	SDIS: Stream disbable: 1 Streaming is disabled - helpful to avoid overrun/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

Bit	R/W	Reset	Description
			3 = HOST_MODE

21.5.8.29 USB2_CONTROLLER_2_USB2D_ENDPTNAK_0

USB2D Endpoint NAK register

Offset: 200h | 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

21.5.8.30 USB2_CONTROLLER_2_USB2D_ENDPTNAK_ENABLE_0

USB2D Endpoint NAK Enable register

Offset: 204h | 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

21.5.8.31 USB2_CONTROLLER_2_USB2D_ENDPTSETUPSTAT_0

USB2D Endpoint Setup Status Register

Offset: 208h | Read/Write: R/W | Reset: 0b0000000000000000

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

Bit	Reset	Description
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 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

Bit	Reset	Description
		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
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

21.5.8.32 USB2_CONTROLLER_2_USB2D_ENDPTPRIME_0

USB2D Endpoint Initialization Register

Offset: 20ch | 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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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

21.5.8.33 USB2_CONTROLLER_2_USB2D_ENDPTFLUSH_0

USB2D Endpoint De-Initialization Register

Offset: 210h | 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

Bit	Reset	Description
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
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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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

21.5.8.34 USB2_CONTROLLER_2_USB2D_ENDPTSTATUS_0

USB2D Endpoint Status Register

Offset: 214h | 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

Bit	Reset	Description
		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
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

Bit	Reset	Description
		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
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
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

Bit	Reset	Description
		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

21.5.8.35 USB2_CONTROLLER_2_USB2D_ENDPTCOMPLETE_0

USB2D Endpoint Complete Register

Offset: 218h | 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
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)

Bit	Reset	Description
		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
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

Bit	Reset	Description
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
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

Bit	Reset	Description
		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

21.5.8.36 USB2_CONTROLLER_2_USB2D_ENDPTCTRL0_0

USB2D Endpoint Control 0 Register

Offset: 21ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

Bit	Reset	Description
		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

21.5.8.37 USB2_CONTROLLER_2_USB2D_ENDPTCTRL1_0

USB2D Endpoint Control 1 Register

Offset: 220h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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.

Bit	R/W	Reset	Description
			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 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

21.5.8.38 USB2_CONTROLLER_2_USB2D_ENDPTCTRL2_0

USB2D Endpoint Control 2 Register

Offset: 224h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			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 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

21.5.8.39 USB2_CONTROLLER_2_USB2D_ENDPTCTRL3_0

USB2D Endpoint Control 3 Register

Offset: 228h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.8.40 USB2_CONTROLLER_2_USB2D_ENDPTCTRL4_0

USB2D Endpoint Control 4 Register

Offset: 22ch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.8.41 USB2_CONTROLLER_2_USB2D_ENDPTCTRL5_0

USB2D Endpoint Control 5 Register

Offset: 230h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.8.42 USB2_CONTROLLER_2_USB2D_ENDPTCTRL6_0

USB2D Endpoint Control 6 Register

Offset: 234h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.8.43 USB2_CONTROLLER_2_USB2D_ENDPTCTRL7_0

USB2D Endpoint Control 7 Register

Offset: 238h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.8.44 USB2_CONTROLLER_2_USB2D_ENDPTCTRL8_0

USB2D Endpoint Control 8 Register

Offset: 23ch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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.

Bit	R/W	Reset	Description
			0 = EP_OK 1 = EP_STALL

21.5.8.45 USB2_CONTROLLER_2_USB2D_ENDPTCTRL9_0

USB2D Endpoint Control 9 Register

Offset: 240h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 is fixed to 0.

Bit	R/W	Reset	Description
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

21.5.8.46 USB2_CONTROLLER_2_USB2D_ENDPTCTRL10_0

USB2D Endpoint Control 10 Register

Offset: 244h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			1 = ISO 2 = BULK 3 = INTR
1	RO	X	RXD: This 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

21.5.8.47 USB2_CONTROLLER_2_USB2D_ENDPTCTRL11_0

USB2D Endpoint Control 11 Register

Offset: 248h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			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 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

21.5.8.48 USB2_CONTROLLER_2_USB2D_ENDPTCTRL12_0

USB2D Endpoint Control 12 Register

Offset: 24ch | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			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 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

21.5.8.49 USB2_CONTROLLER_2_USB2D_ENDPTCTRL13_0

USB2D Endpoint Control 13 Register

Offset: 250h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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 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

21.5.8.50 USB2_CONTROLLER_2_USB2D_ENDPTCTRL14_0

USB2D Endpoint Control 14 Register

Offset: 254h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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 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

Bit	R/W	Reset	Description
			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 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

21.5.8.51 USB2_CONTROLLER_2_USB2D_ENDPTCTRL15_0

USB2D Endpoint Control 15 Register

Offset: 258h | Read/Write: R/W | Reset: 0b000x00x0xxxxxxxx000x00x0

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

Bit	R/W	Reset	Description
			3 = INTR
17	RO	X	TXD: This 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 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

21.5.9 USB3 Controller Interface Registers

These are used to generate the actual RTL registers for USB3 controller interface.

21.5.9.1 USB3_IF_USB_SUSP_CTRL_0

USB suspend control register

This register controls the suspend and reusume behavior of USB controller/PHY.

Offset: 400h | Read/Write: R/W | Reset: 0b00xxxxxx0000x001000xx000000

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

Bit	R/W	Reset	Description
			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.
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
13	RW	0x0	ICUSB_PHY_ENB: Enable ICUSB PHY mode Setting this will enable the PLLU output clock when ICUSB is not in suspend 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
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 USB PHY clock is waked 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 wakeup 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 wakeup 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 wakeup 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

21.5.9.2 USB3_IF_USB_PHY_VBUS_SENSORS_0

USB PHY VBUS SENSORS control register

This register controls the OTG VBUS sensors in the USB PHY. We have the following 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 _STS. The field _CHG_DET is set to 1 whenever a change is detected in the value of . 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 , it can set _SW_EN to 1, and set _SW_VALUE to 1 or 0 as per the requirement.

We have 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 approrpriate 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.

Note: The source for A_SESS_VLD sensor can be programmed according to the setting of USB1_VBUS_SENSE_CTL in register USB1_LEGACY_CTRL.

Offset: 404h | Read/Write: R/W | Reset: 0b0000x00x0000x00x0000x00x0000x00

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 wakeup 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 wakeup 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

Bit	R/W	Reset	Description
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 wakeup 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 wakeup 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: 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.

Bit	R/W	Reset	Description
			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

21.5.9.3 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. We have the following sensors in this register:

- VBUS_WAKEUP
- ID
- VDAT_DET
- VDCD_DET

The debounced status of each sensor can be read from _STS. The field _CHG_DET is set to 1 whenever a change is detected in the value of . 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 , it can set _SW_EN to 1, and set _SW_VALUE to 1 or 0 as per the requirement.

We have 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 apprORpriate 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 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: 408h | Read/Write: R/W | Reset: 0b0000x00xx000x00xx000x00x1000x00

Bit	R/W	Reset	Description
30	RW	0x0	VBUS_WAKEUP_WAKEUP_EN: VBUS_WAKEUP wakeup enable. If this bit is enabled, USB will wakeup 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 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.

Bit	R/W	Reset	Description
			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
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

21.5.9.4 USB3_IF_USB_PHY_ALT_VBUS_STS_0

USB PHY Alternate VBUS status register

Offset: 40ch | Read/Write: RO | Reset: 0bxxxxxxxxx

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

21.5.9.5 USB3_IF_ICUSB_XCVR_CFG_0

ICUSB Transceiver Configuration register

Offset: 414h | 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

Bit	R/W	Reset	Description
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

21.5.9.6 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 as device never transmits two packets in a row.

Offset: 420h | Read/Write: R/W | Reset: 0b010010

Bit	Reset	Description
5:0	0x12	IP_DELAY_TX2TX_HS: HS Tx to Tx inter-packet delay. This is valid only for UTMIP PHY Software should not change this.

21.5.9.7 USB3_IF_USB_RSM_DLY_0

USB Controller Resume Signalling Delay

Offset: 490h | Read/Write: R/W | Reset: 0b0110100101111000

Bit	Reset	Description
15:0	0x6978	TIME_TO_RESUME: Send the resume back in no. of 60 MHz cycles. Default gives 900 usec delay. Only applicable in host mode.

21.5.9.8 USB3_IF_ICUSB_PADCTLS_0

ICUSB Pad Controls Config

Offset: 494h | Read/Write: R/W | Reset: 0b000000000

Bit	Reset	Description
8	0x0	ICUSB_RPU2_EN: Config RPU2_EN state. 0 = DISABLE 1 = ENABLE
7	0x0	ICUSB_DISABLE_PULLUP_DP: Force DP pullup inactive. (Overrides FORCE_PULLUP_DP.)
6	0x0	ICUSB_DISABLE_PULLUP_DM: Force DM pullup inactive. (Overrides FORCE_PULLUP_DM.)
5	0x0	ICUSB_DISABLE_PULLDN_DP: Force DP pulldown inactive. (Overrides FORCE_PULLDN_DP.)
4	0x0	ICUSB_DISABLE_PULLDN_DM: Force DM pulldown inactive. (Overrides FORCE_PULLDN_DM.)

Bit	Reset	Description
3	0x0	ICUSB_FORCE_PULLUP_DP: Force DP pullup active.
2	0x0	ICUSB_FORCE_PULLUP_DM: Force DM pullup active.
1	0x0	ICUSB_FORCE_PULLDN_DP: Force DP pulldown active.
0	0x0	ICUSB_FORCE_PULLDN_DM: Force DM PullDown active.

21.5.9.9 USB3_IF_SPARE_0

Spare Register

Offset: 498h | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:16	0xffff	SPARE_HI: Spare register bits, defaulted to 1
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

21.5.10 USB 3 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

21.5.10.1 USB3_UTMIP_PLL_CFG0_0

USB_PHY PLL Configuration Register 0 The data sampling frequency relies on a 960MHz clock, so the goal of the PLL is to have: In_Frequency*(PLL_VCOMULTBY2+1)*(PLL_NDIV/PLL_MDIV) = 960MHz With a 12MHz input from PLL_U, the default setting of PLL_VCOMULTBY2=1, PLL_NDIV=40 and PLL_MDIV=1 results in a correct output.

Note: This register has been moved to the CAR section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT WHATSOEVER!

This register was used to configure PLL inside utmip block prior to Tegra 3. 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.

Offset: 800h | Read/Write: R/W | Reset: 0b00000000010100000000000110000000

Bit	Reset	Description
30:28	0x0	UTMIP_PLL_SELECT: Selects which of the eight 60MHz clock phases to produce at the output of the USB PHY PLL. This is for a test mode. See cell specification.
27	0x0	UTMIP_PLL_PDIVRST: PDIVRST of the UTMIP PHY PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.
26:24	0x0	UTMIP_PLL_PDIV: PDIV[2:0] input of the USB_PHY PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.
23:16	0x28	UTMIP_PLL_NDIV: NDIV[7:0] input of USB_PHY 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 USB_PHY PLL. This is the predivide on the PLL. 0x0 is not allowed. See cell specification.
7	0x1	UTMIP_PLL_VCOMULTBY2: VCOMULTBY2 control of the UTMIP PHY PLL. Recommended setting is

Bit	Reset	Description
		on. Additional divide by 2 on the VCO feedback. Which is setting the bit to 1. See cell spec.
6:1	0x0	UTMIP_PLL_LOCKSEL: LOCKSEL[5:0] input of USB_PHY PLL. Used in test modes only. See cell specification.
0	0x0	UTMIP_PLL_LOCKENABLE: LOCK_ENABLE input of USB_PHY PLL. Normally only used during test. See cell specification.

21.5.10.2 USB3_UTMIP_PLL_CFG1_0

UTMIP PLL and PLLU configuration register 1

Note: This register has been moved to the CAR section of this document. PROGRAMMING THIS REGISTER HERE WILL HAVE NO EFFECT WHATSOEVER!

This register was used to configure PLL inside utmip block prior to Tegra 3. This has been de-featured from utmip space and moved to car space.

PLL CONFIGURATION & PARAMETERS

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

Xtal clock -> enters PLL_U to generate 12MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock

The following parameters control the bringup of the plls:

Coming out of reset or suspend

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

pll_active_count = 0

- -> Wait 5us to enable pll_active: pll_active_count == ClkXtal * PLL_ACTIVE_DLY_COUNT * 16 => USB_PHY PLL_ACTIVE
- -> Wait ~2.5ms until USB_PHY is actually stable (pll_lock_count == ClkXtal * XTAL_FREQ_COUNT * 256) => USB_PHY

Numbers for a 19.2MHz Xtal 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)
- PLL_ACTIVE_DLY_COUNT[4:0] = 5 * 19.2 / 16 = 6 = 0x06
- XTAL_FREQ_COUNT[11:0] = 2500 * 19.2 / 256 = 187 = 0xbb

Offset: 804h | Read/Write: R/W | Reset: 0b00011000001000000000000011000000

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	0x8	UTMIP_PLL_SETUP: SETUP[8:0] input of USB_PHY PLL.
17	0x0	UTMIP_FORCE_PLLU_POWERUP: Force PLL_U into power up.

Bit	Reset	Description
16	0x0	UTMIP_FORCE_PLLU_POWERDOWN: Force PLL_U into power down. (Overrides FORCE_PLLU_POWERUP.)
15	0x0	UTMIP_FORCE_PLL_ENABLE_POWERUP: Force USB_PHY PLL pll_enable input on.
14	0x0	UTMIP_FORCE_PLL_ENABLE_POWERDOWN: Force USB_PHY PLL pll_enable input off. (Overrides FORCE_PLL_ENABLE_POWERUP.)
13	0x0	UTMIP_FORCE_PLL_ACTIVE_POWERUP: Force USB_PHY PLL pll_active input on.
12	0x0	UTMIP_FORCE_PLL_ACTIVE_POWERDOWN: Force USB_PHY 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 USB_PHY PLL is considered stable.

21.5.10.3 USB3_UTMIP_XCVR_CFG0_0

UTMIP transceiver cell configuration register 0

Offset: 808h | 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.)
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.

21.5.10.4 USB3_UTMIP_BIAS_CFG0_0

UTMIP Bias cell configuration register 0

Offset: 80ch | Read/Write: R/W | Reset: 0b01100000000000110000000000

Bit	Reset	Description
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.
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.

21.5.10.5 USB3_UTMIP_HSRX_CFG0_0

UTMIP High speed receive config 0

Offset: 810h | 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 cycles of idle 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

Bit	Reset	Description
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

21.5.10.6 USB3_UTMIP_HSRX_CFG1_0

UTMIP High speed receive config 1

Offset: 814h | Read/Write: R/W | Reset: 0b010011

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

21.5.10.7 USB3_UTMIP_FSLSRX_CFG0_0

UTMIP full and Low speed receive config 0

Offset: 818h | Read/Write: R/W | Reset: 0b11111101010101001000010000101001

Bit	Reset	Description
31	0x1	UTMIP_FSLS_SE1_DRIBBLE_FILTER: One SE1, don't allow dribble
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 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

21.5.10.8 USB3_UTMIP_FSLSRX_CFG1_0

UTMIP full and Low speed receive config 1

Offset: 81ch | Read/Write: R/W | Reset: 0b01000100110011101010000000000

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

21.5.10.9 USB3_UTMIP_TX_CFG0_0

UTMIP transmit config signals

Offset: 820h | Read/Write: R/W | Reset: 0b00010000001000000000

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
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

Bit	Reset	Description
0	0x0	UTMIP_NO_SYNC_NO_EOP: Do not sent SYNC or EOP

21.5.10.10 USB3_UTMIP_MISC_CFG0_0

UTMIP miscellaneous configurations

Offset: 824h | Read/Write: R/W | Reset: 0b00000111110000000000000001111000

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: Don't block changes for 4ms 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.
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: Don't 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

21.5.10.11 USB3_UTMIP_MISC_CFG1_0

UTMIP miscellaneous configurations

Offset: 828h | Read/Write: R/W | Reset: 0b10000000001100110000000000100100

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 neg edge sync for linestate
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

21.5.10.12 USB3_UTMIP_DEBOUNCE_CFG0_0

UTMIP Avalid and Bvalid debounce

Debounce values IdDig, Avalid, Bvalid, VbusValid, VbusWakeUp, and SessEnd. Each of these signals have their 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, we have: BIAS_DEBOUNCE_A[15:0] = 1000 * 19.2 / 4 = 4800 = 0x12c0

Offset: 82ch | 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

21.5.10.13 USB3_UTMIP_BAT_CHRG_CFG0_0

UTMIP battery charger configuration

Offset: 830h | Read/Write: R/W | Reset: 0b000000xx000001

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

21.5.10.14 USB3_UTMIP_SPARE_CFG0_0

Utmip spare configuration bits

Offset: 834h | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:0	-65536	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 31 to 5: Reserved Note: 4:3 stand for using fused termination and setup value.

21.5.10.15 USB3_UTMIP_XCVR_CFG1_0

UTMIP transceiver cell configuration register 1

Offset: 838h | Read/Write: R/W | Reset: 0b00011000100000010000010101

Bit	Reset	Description
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

Bit	Reset	Description
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 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 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.)

21.5.10.16 USB3_UTMIP_BIAS_CFG1_0

UTMIP Bias cell configuration register 1

Offset: 83ch | Read/Write: R/W | Reset: 0b00000000101101

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 20us. For a Xtal clock of 13MHz it should be set a 5.
2	0x1	UTMIP_VBUS_WAKEUP_POWERDOWN: Force VBUS_WAKEUP input into power down. 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.)

21.5.10.17 USB3_UTMIP_BIAS_STS0_0

UTMIP Bias cell status register 0

Offset: 840h | 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.

21.5.10.18 USB3_UTMIP_CHRG_DEB_CFG0_0

UTMIP VDcd_Det and VDat_Det debounce

Debounce values VDcd_Det and VDat_Det. Each of these signals have their own debouncer and 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: 844h | 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

21.5.10.19 USB3_UTMIP_MISC_STS0_0

UTMIP MISC status reg

Offset: 848h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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

21.5.10.20 USB3_UTMIP_PMC_WAKEUP0_0

UTMIP PMC Wakeup value

Offset: 84ch | Read/Write: R/W | Reset: 0bx0

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

21.5.10.21 USB2_QH_USB2D_QH_EP_0_OUT_0

USB2D Queue Head for OUT endpoint 0

Offset: 1000h | 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.

21.5.10.22 USB2_QH_USB2D_QH_EP_0_IN_0

USB2D Queue Head for IN endpoint 0

Offset: 1040h | 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.

21.5.10.23 USB2_QH_USB2D_QH_EP_1_OUT_0

USB2D Queue Head for OUT endpoint 1

Offset: 1080h | 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.

21.5.10.24 USB2_QH_USB2D_QH_EP_1_IN_0

USB2D Queue Head for IN endpoint 1

Offset: 10c0h | 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.

21.5.10.25 USB2_QH_USB2D_QH_EP_2_OUT_0

USB2D Queue Head for OUT endpoint 2

Offset: 1100h | 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.

21.5.10.26 USB2_QH_USB2D_QH_EP_2_IN_0

USB2D Queue Head for IN endpoint 2

Offset: 1140h | 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.

21.5.10.27 USB2_QH_USB2D_QH_EP_3_OUT_0

USB2D Queue Head for OUT endpoint 3

Offset: 1180h | 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.

21.5.10.28 USB2_QH_USB2D_QH_EP_3_IN_0

USB2D Queue Head for IN endpoint 3

Offset: 11c0h | 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.

21.5.10.29 USB2_QH_USB2D_QH_EP_4_OUT_0

USB2D Queue Head for OUT endpoint 4

Offset: 1200h | 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.

21.5.10.30 USB2_QH_USB2D_QH_EP_4_IN_0

USB2D Queue Head for IN endpoint 4

Offset: 1240h | 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.

21.5.10.31 USB2_QH_USB2D_QH_EP_5_OUT_0

USB2D Queue Head for OUT endpoint 5

Offset: 1280h | 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.

21.5.10.32 USB2_QH_USB2D_QH_EP_5_IN_0

USB2D Queue Head for IN endpoint 5

Offset: 12c0h | 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.

21.5.10.33 USB2_QH_USB2D_QH_EP_6_OUT_0

USB2D Queue Head for OUT endpoint 6

Offset: 1300h | 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.

21.5.10.34 USB2_QH_USB2D_QH_EP_6_IN_0

USB2D Queue Head for IN endpoint 6

Offset: 1340h | 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.

21.5.10.35 USB2_QH_USB2D_QH_EP_7_OUT_0

USB2D Queue Head for OUT endpoint 7

Offset: 1380h | 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.

21.5.10.36 USB2_QH_USB2D_QH_EP_7_IN_0

USB2D Queue Head for IN endpoint 7

Offset: 13c0h | 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.

21.5.10.37 USB2_QH_USB2D_QH_EP_8_OUT_0

USB2D Queue Head for OUT endpoint 8

Offset: 1400h | 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.

21.5.10.38 USB2_QH_USB2D_QH_EP_8_IN_0

USB2D Queue Head for IN endpoint 8

Offset: 1440h | 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.

21.5.10.39 USB2_QH_USB2D_QH_EP_9_OUT_0

USB2D Queue Head for OUT endpoint 9

Offset: 1480h | 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.

21.5.10.40 USB2_QH_USB2D_QH_EP_9_IN_0

USB2D Queue Head for IN endpoint 9

Offset: 14c0h | 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.

21.5.10.41 USB2_QH_USB2D_QH_EP_10_OUT_0

USB2D Queue Head for OUT endpoint 10

Offset: 1500h | 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.

21.5.10.42 USB2_QH_USB2D_QH_EP_10_IN_0

USB2D Queue Head for IN endpoint 10

Offset: 1540h | 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.

21.5.10.43 USB2_QH_USB2D_QH_EP_11_OUT_0

USB2D Queue Head for OUT endpoint 11

Offset: 1580h | 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.

21.5.10.44 USB2_QH_USB2D_QH_EP_11_IN_0

USB2D Queue Head for IN endpoint 11

Offset: 15c0h | 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.

21.5.10.45 USB2_QH_USB2D_QH_EP_12_OUT_0

USB2D Queue Head for OUT endpoint 12

Offset: 1600h | 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.

21.5.10.46 USB2_QH_USB2D_QH_EP_12_IN_0

USB2D Queue Head for IN endpoint 12

Offset: 1640h | 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.

21.5.10.47 USB2_QH_USB2D_QH_EP_13_OUT_0

USB2D Queue Head for OUT endpoint 13

Offset: 1680h | 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.

21.5.10.48 USB2_QH_USB2D_QH_EP_13_IN_0

USB2D Queue Head for IN endpoint 13

Offset: 16c0h | 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.

21.5.10.49 USB2_QH_USB2D_QH_EP_14_OUT_0

USB2D Queue Head for OUT endpoint 14

Offset: 1700h | 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.

21.5.10.50 USB2_QH_USB2D_QH_EP_14_IN_0

USB2D Queue Head for IN endpoint 14

Offset: 1740h | 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.

21.5.10.51 USB2_QH_USB2D_QH_EP_15_OUT_0

USB2D Queue Head for OUT endpoint 15

Offset: 1780h | 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.

21.5.10.52 USB2_QH_USB2D_QH_EP_15_IN_0

USB2D Queue Head for IN endpoint 15

Offset: 17c0h | 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.

21.5.11 CLK_RST_CONTROLLER_UTMIP_PLL_CFG0_0

This register is also a part of the CAR section of this document.

The data sampling frequency relies on a 960MHz clock, so the goal of the PLL is to have: In_Frequency*(PLL_VCOMULTBY2+1)*(PLL_NDIV/PLL_MDIV) = 960MHz With a 12MHz 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: 480h | Read/Write: R/W | Reset: 0b00000000010100000000000110000000

Bit	Reset	Description
30:28	0x0	UTMIP_PLL_SELECT: Selects which of the eight 60MHz clock phases to produce at the output of the USB PHY PLL. This is for a test mode. See cell specification.
27	0x0	UTMIP_PLL_PDIVRST: PDIVRST of the UTMIP PHY PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.
26:24	0x0	UTMIP_PLL_PDIV: PDIV[2:0] input of the UTMIP PLL. Reserved. Keep at 0x0. Currently there is no post divider on this PLL. See cell specification.
23:16	0x28	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.
7	0x1	UTMIP_PLL_VCOMULTBY2: VCOMULTBY2 control of the UTMIP PHY PLL. Recommended setting is on. Additional divide by 2 on the VCO feedback. Which is setting the bit to 1. See cell spec.
6:1	0x0	UTMIP_PLL_LOCKSEL: LOCKSEL[5:0] input of UTMIP PLL. Used in test modes only. See cell specification.
0	0x0	UTMIP_PLL_LOCKENABLE: LOCK_ENABLE input of UTMIP PLL. Normally only used during test. See cell specification.

21.5.12 CLK_RST_CONTROLLER_UTMIP_PLL_CFG1_0

PLL Configuration and Parameters

This register is also a part of the CAR section of this document.

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

Xtal clock -> enters PLL_U to generate 12MHz clock -> enters USB_PHY PLL to generate 480/60 MHz clock

The following parameters control the bringup of the plls:

Coming out of reset or suspend

-> PllUOnState: start pllu_enable_count and pll_lock_count

-> Wait ~1us to actually enable the PLL_U (pllu_enable_count == ClkXtal * PLLU_ENABLE_DLY_COUNT * 8)

-> Wait ~1ms until PLL_U is actually stable (pll_lock_count == ClkXtal * PLLU_STABLE_COUNT * 256) => USB_PHY PLL_ENABLE pll_active_count = 0

-> Wait 5us to enable pll_active: pll_active_count == ClkXtal * PLL_ACTIVE_DLY_COUNT * 16 => USB_PHY PLL_ACTIVE

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

Numbers for a 19.2MHz Xtal 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
- PLL_ACTIVE_DLY_COUNT[4:0] = 10 * 19.2 / 16 = 12 = 0x0c
- XTAL_FREQ_COUNT[11:0] = 2500 * 19.2 / 256 = 187 = 0xbb

UTMIP PLL and PLLU configuration register 1

Offset: 484h | Read/Write: R/W | Reset: 0b00011000001000010101000011000000

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	0x8	UTMIP_PLL_SETUP: SETUP[8:0] input of USB_PHY PLL.
17	0x0	UTMIP_FORCE_PLLU_POWERUP: Force PLL_U into power up.
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.

21.5.13 CLK_RST_CONTROLLER_UTMIP_PLL_CFG2_0

This register is also a part of the CAR section of this document.

UTMIP Miscellaneous Configurations

Offset: 488h | Read/Write: R/W | Reset: 0b10101xx00110001100000000010101

Bit	Reset	Description
30	0x1	UTMIP_PHY_XTAL_CLOCKEN: Selects whether to enable the crystal clock in the module.
29	0x0	UTMIP_FORCE_PD_CLK60_POWERUP: Force UTMIP PLL PD_CLK60 input into power up.
28	0x1	UTMIP_FORCE_PD_CLK60_POWERDOWN: Force UTMIP PLL PD_CLK60 input into power down. (Overrides FORCE_PD_CLK60_POWERUP.)
27	0x0	UTMIP_FORCE_PD_CLK48_POWERUP: Force UTMIP PLL PD_CLK48 input into power up.
26	0x1	UTMIP_FORCE_PD_CLK48_POWERDOWN: Force UTMIP PLL PD_CLK48 input into power down. (Overrides FORCE_PD_CLK48_POWERUP.)
23:18	0xc	UTMIP_PLL_ACTIVE_DLY_COUNT: 10 us / (1/19.2MHz) = 192 / 16 = 12
17:6	0x600	UTMIP_PLLU_STABLE_COUNT: WRONG! This should be 1ms -> 0x50
5	0x0	UTMIP_FORCE_PD_SAMP_C_POWERUP: Force UTMIP PLL PD_SAMP_C input into power up.

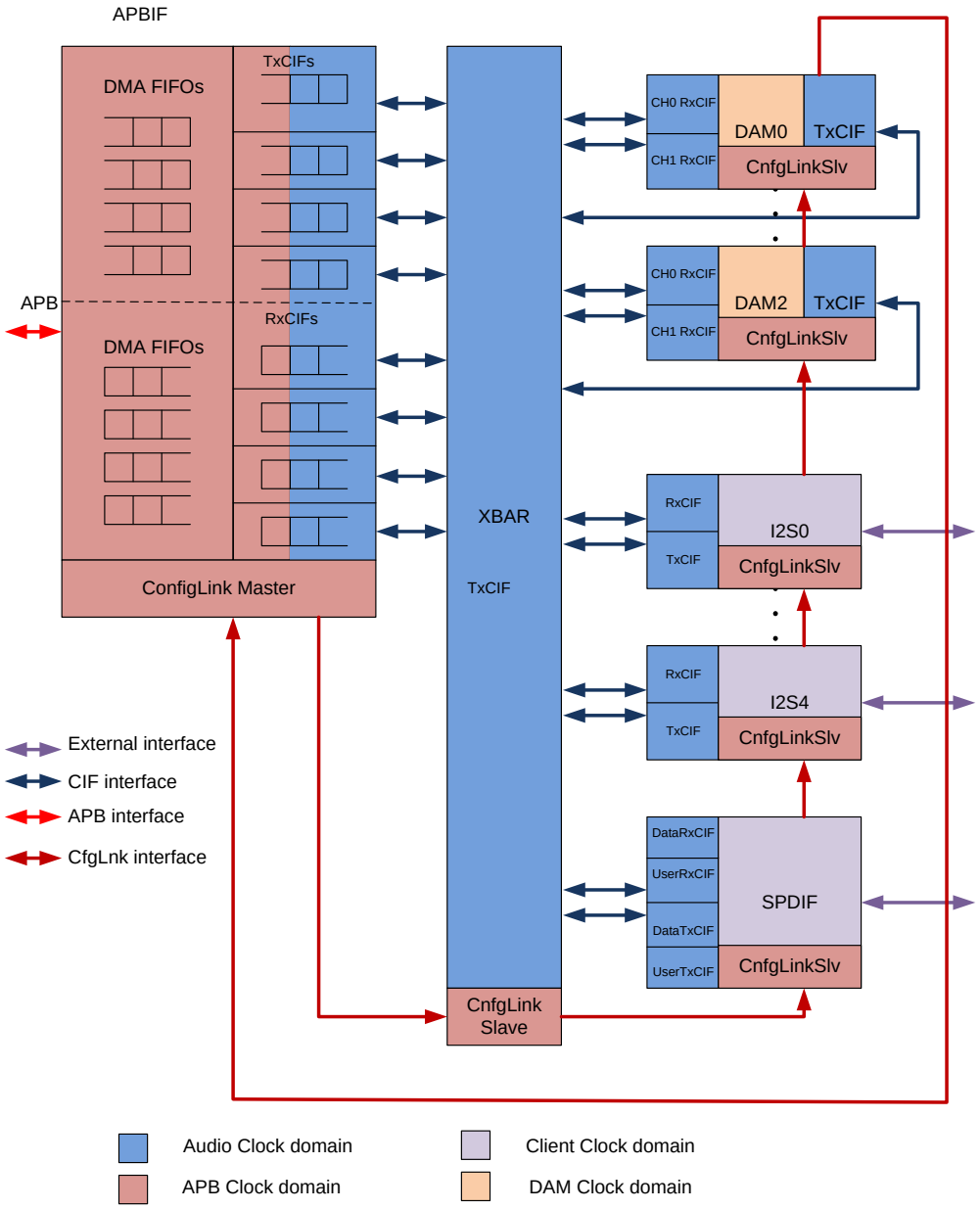
Bit	Reset	Description
4	0x1	UTMIP_FORCE_PD_SAMP_C_POWERDOWN: Force UTMIP PLL PD_SAMP_C input into power down. (Overrides FORCE_PD_SAMP_C_POWERUP.)
3	0x0	UTMIP_FORCE_PD_SAMP_B_POWERUP: Force UTMIP PLL PD_SAMP_B input into power up.
2	0x1	UTMIP_FORCE_PD_SAMP_B_POWERDOWN: Force UTMIP PLL PD_SAMP_B input 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.)

22.0 AUDIO HUB

The Tegra® 3 processor defines several audio interface modules to support various types of audio protocols and data formats. To provide a flexible audio streaming environment, it is essential to implement a versatile audio fabric to connect audio modules independently and simultaneously.

The Audio Hub (AHUB) is a crossbar switch matrix connecting various audio modules such as I2S and SPDIF. AHUB is attached to the APB bus and is programmable through the bus. APBIF is the agent for the APB control flow and DMA operation, which sends or receives data from/to Memory. The figure below shows the audio subsystem in Tegra 3 devices.

Figure 38. Audio Subsystem in Tegra 3 Devices



Features

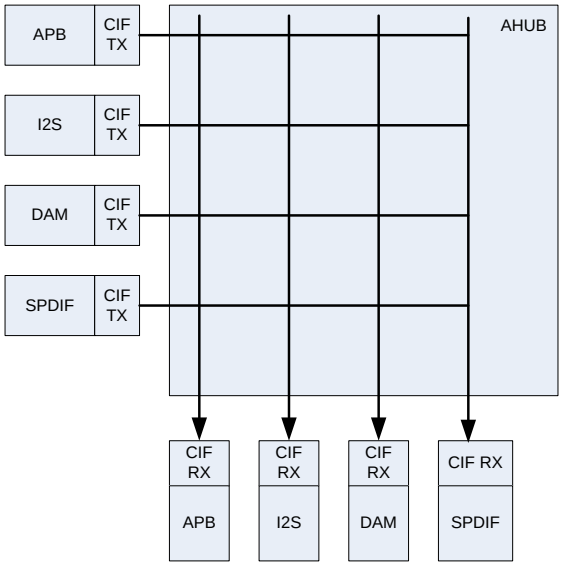
- Concurrent switching between audio clients
  - The audio clients are I2S, DAM, SPDIF, APBIF
  - A TX client can talk to multiple RX clients simultaneously
- Scalable MxN crossbar switch, where
  - M is the number of TX clients
  - N is the number of RX clients
- A native protocol, AHUB protocol, is defined and used between AHUB and AHUB clients
- Programmable clock
  - Typically 12 MHz from crystal
- Supports multi-channel APB DMA operations
- AHUBCIF provides the interface to AHUB in the AHUB clients
  - One or more AHUBCIF can be instantiated in an AHUB client for RX and TX
  - Data conversion between various bit formats, 8/16/24/32
  - Mono/Stereo conversion
  - Asynchronous clock boundary between AHUB clock and client clock

22.1 Audio Hub Functionality

AHUB is a 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 a 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. Figure 39 shows a simple diagram of the AHUB crossbar switch.

One or more AHUBCIFs are instantiated in an AHUB client to transmit and/or receive audio data through AHUB. The AHUBCIF generate audio data frames with the AHUB protocol, which will be discussed later. It also performs a common data conversion in most of the clients, data width conversion (8/16/24/32) and channel conversion (mono/stereo).

Figure 39. AHUB Crossbar Switch and Clients



22.1.1 AHUB signals

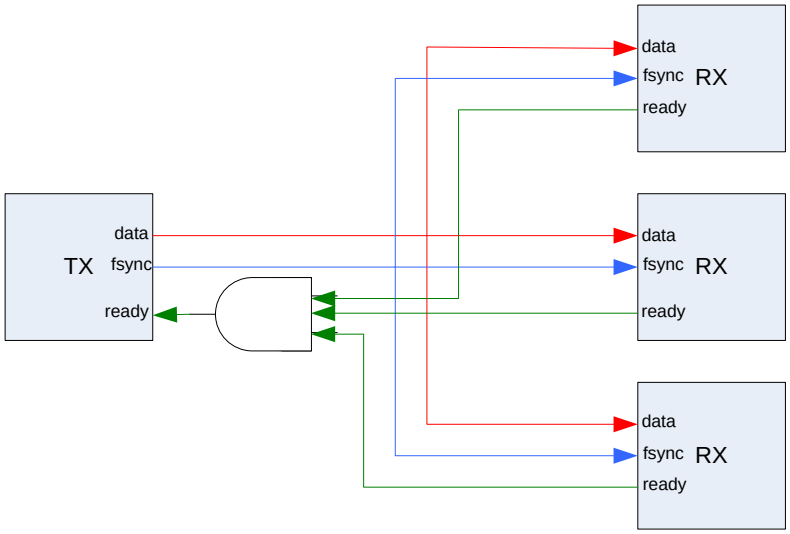
An AHUB session has 3 signals to transmit audio data from a TX client to RX clients. Table 37 shows the signals. Figure 40 shows the connectivity between the Tx and Rx clients through the AHUB.

The data signal transmits one bit each clock. The fsync signal is asserted when a new frame starts. A frame consists of samples from multiple channels. When a RX client asserts the Ready signal, a buffer of at least one frame should be ready in the RX client's AHUBCIF. The Tx client samples 'Ready' only when the fsync is asserted. If the ready signal is de-asserted when the fsync is active (indicting that the Rx client is not ready to receive data), the TX client should hold the first data and fsync until the Ready signal is asserted.

Table 37. AHUB signals

signal	Width	direction	Description
data	1	TX->RX	serial audio data
fsync	1	TX->RX	a pulse frame sync for each frame
ready	1	TX<-RX	Ready from RX clients

Figure 40. Signals exchange between TX and RX clients



22.1.2 Frame Length

The frame length is simply a product of bits per sample and the number of channels in an audio frame. Before starting data transfer, Bits per sample and the number of channels of the Tx and Rx clients should be programmed to be identical, to avoid incorrect frame parsing. Table 38 show the min and max length of frames that AHUB carries.

Table 38. Frame length

	Bits	channels	frame length	typical usecase
min frame	8	1	8	mono voice stream from baseband
max frame	24	8	192	HIFI multi-channel audio in HDMI

22.1.3 Clocking

When multiple sessions are in progress, AHUB clock should be fast enough to support the session with the highest data rate. If a session is used for sending 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 application like MP3 music playback. Depending on the applications, you can program the clock registers to satisfy the requirement of clock frequency. Table 39 shows the various clock frequencies for AHUB.

Table 39. AHUB Clock Frequencies

clock freq	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 48k Hz 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 max)

22.1.4 AHUBCIF

The main function of AHUBCIF is to serialize an audio sample to a serial data stream or de-serialize 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. An asynchronous FIFO is used to cross the clock boundary between *ahub_clk* and *client_clk*.

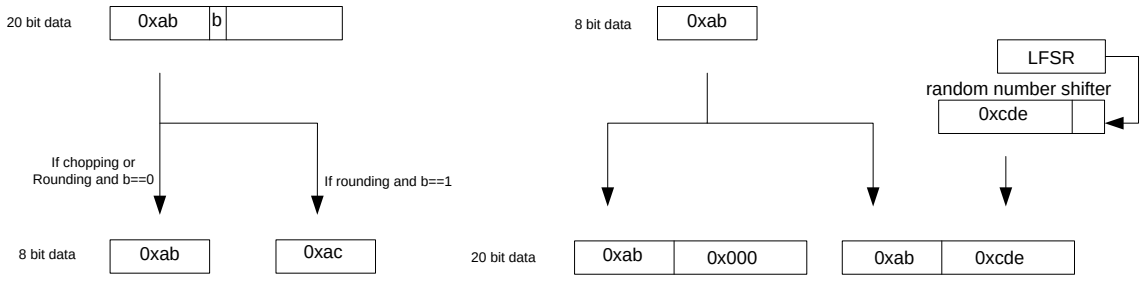
Besides serialization/de-serialization, other common format conversions performed in AHUBCIF optionally by the register configuration include data width conversion and channel conversion.

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. AHUBCIF maintains a 16 bit LSFR to generate random numbers, x**16 + x**12 + x**5 + 1.

LFSR = ((LFSR << 1) ^ ((LFSR & 0x8000)? 0x1021: 0x0000)) & 0xFFFF

Figure 41 shows how AHUBCIF converts different widths of data with randomly generated numbers.

Figure 41. 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.

While 16/24 bit PCM samples are represented by 2's complements, 8 bit samples are different.

8 bit sample	actual value
0xFF	127
0x80	0
0x00	-128

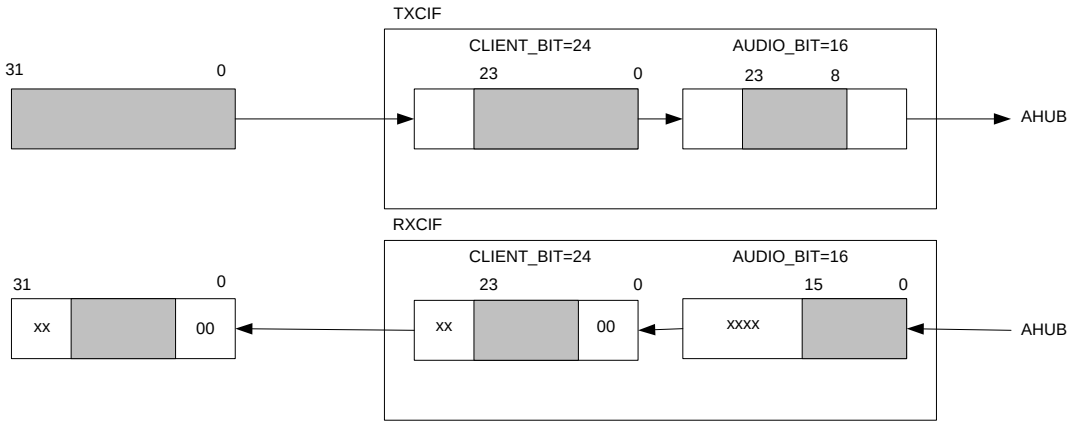
When an 8 bit sample A[7:0] is extended into a 16 bit sample B[15:0],

B[15:8] = 0x80 ^ A[7:0]

B[7:0] = 0's or 1's or 8 bit LFSR

When a 16 bit sample is converted to an 8 bit sample, the process should be reversed.

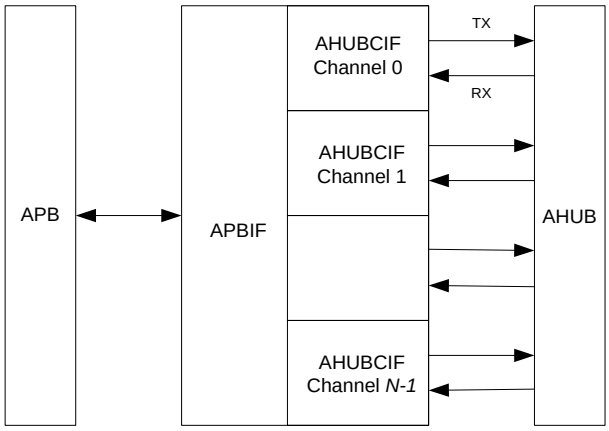
Bit width of audio sample on AHUB side is specified by AUDIO_BITS field. Bit width of audio samples in client domain is specified by CLIENT_BITS field. Since AHUBCIF has 32 bit data interface, the maximum value of CLIENT_BITS and AUDIO_BITS is 32. When the bus is defined to be DATA[31:0], CLIENT_BITS 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 is 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.



22.2 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 42. 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

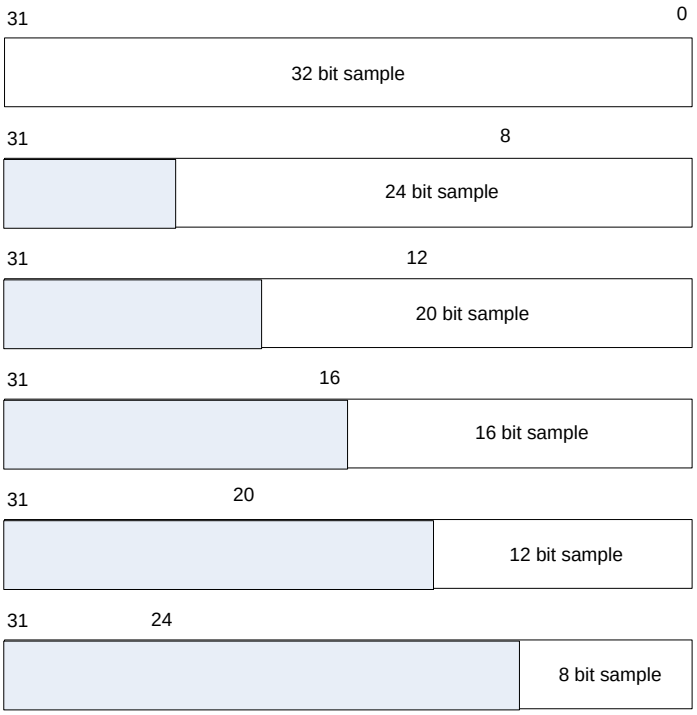
22.2.1 APBIF Functionality

22.2.1.1 Number of channels

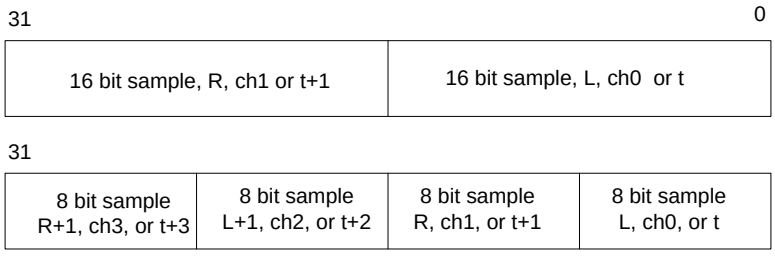
Tegra 3 supports 4 full duplex DMA channels.

22.2.1.2 Data Packing

When an audio sample is saved in the memory, it is right justified in a D-word 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, 2x16bits and 4x8bits 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 4x8bits configuration, the one in the LSB is the earlier left channel sample or the earliest sample in case of single channel stream.



22.3 Digital Audio Mixer (DAM)

Tegra® 3 Digital Audio Mixer (DAM) has four inputs and one output. The inputs are from AHUB through AHUBCIF, while the output also goes to AHUB through AHUBCIF. One input is for voice stream, while the others are for high-fidelity audio streams. Typically for the cellular communication, the voice stream comes from a baseband processor through an I2S unit and AHUB. The voice stream is normally up-converted to a high sampling rate, either 44.1 or 48 kHz before it is mixed with another audio stream. The down-conversion is also possible when high sample rate needs to be converted to low sample rate.

Features

- DAM takes two input sources, typically one for voice data from a baseband (BB) and the others for music from an internal source.
  - 2 input sources with 2 RX AHUBCIFs
  - 1 output with 1 TX AHUBCIF
- Sample rate conversion for voice stream
  - One input for Low-Latency-Filter(LLF) SRC

- Low-Latency-Filter(LLF) SRC
  - input: 8/16/44.1/48 kHz mono audio
  - up-conversion from 8/16 -> 44.1/48 kHz
  - down-conversion from 44.1/48 -> 8/16 kHz
  - 16 bit audio data
  - 4th order Farrow filter for interpolation
  - 10th order elliptic filter for anti-aliasing
  - fixed latency on round trip between baseband and CODEC of ~1msec.
- DAM mixes the two audio input through SRC and the programmable gain controller.
- Configuration register programming

22.3.1 DAM Functionality

22.3.1.1 AHUBCIF RX

DAM has 2 AHUBCIF instances for stream inputs. AHUBCIF can perform simple format conversion for bit-width and/or multiple-channels. LLF SRC only handles 16 bit mono stream

22.3.1.2 Low Latency Filter (LLF)

SRC performs up and down sample conversion, which addresses the conversion between 8/16 kHz and 44.1/48 kHz sampling rates. LLF is composed of two pipeline stages, interpolation and anti-aliasing. A Farrow filter is used for interpolation. An IIR filter is used for anti-aliasing.

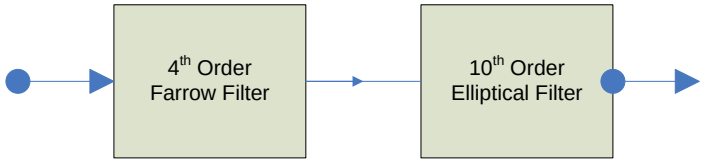
Up sampling

The interpolation filter is applied to generate more samples per unit time and the anti-aliasing filter is applied to sort out unwanted frequency signals. Suppose the input and the output sample rates are M and N kHz, respectively, where M < N.

```
                step_size = S
                current_step = 0x8000;
for each sample (0 .. STEP_RESET-1) {
                calculate step_index and step_frag from current_step
        step_index = (current_step & 0x8000) >> 15
        step_frag = current_step & 0x7FFF
                if(step_index == 1) {
                     pop a sample from the flow control FIFO and add it into Q
                     current_step = current_step - 0x8000;
                }
    apply Farrow Filter with Q and step_frag
    time = time + step_size
    apply Elliptic Filter and generate a sample to output
}
```

The coefficients are generated based on a step size and an index value of the current time. The tables below show the step sizes and index values for up-sampling and down-sampling, respectively.

up-sampling	step index	step fractional(hex)
8kHz -> 44.1kHz	0	0x1738
8kHz -> 48 kHz	0	0x1555
16kHz -> 44.1kHz	0	0x2E70
16kHz -> 48kHz	0	0x2AAA



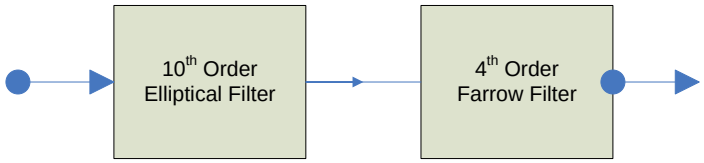
Down Sampling

The anti-aliasing filter is applied first and the interpolation filter is applied to generate less number of samples per unit time. Suppose the input and the output sample rates are M and N kHz, respectively, where M > N.

```
                step_size = S
                current_step = 0x0000
                              for each sample (0 ..  STEP_RESET-1) {
                                  pop a sample from the flow control FIFO, apply Elliptic Filter and add
it into Q
                    calculate step_index and step_frag from current_step
                        step_index = (current_step & 0xF8000) >> 15
                        step_frag = current_step & 0x7FFF
                    if(step_index  == 0) {
                        if(step_flag == 0)
                                send the middle sample in Q to output
                        else
                          apply Farrow Filter with Q and step flag and generate a sample to output
                        current_step  = current_step + step_size
                    }
                    current_step = current_step – 0x8000
                 }
```

The coefficients are generated based on a step size and an index value of the current time. The tables below show the step size and index values for up-sampling and down-sampling, respectively.

down-sampling	step index	step fractional(hex)
8kHz -> 44.1kHz	5	0x4199
8kHz -> 48 kHz	6	0x0
16kHz -> 44.1kHz	2	0x60cc
16kHz -> 48kHz	3	0x0

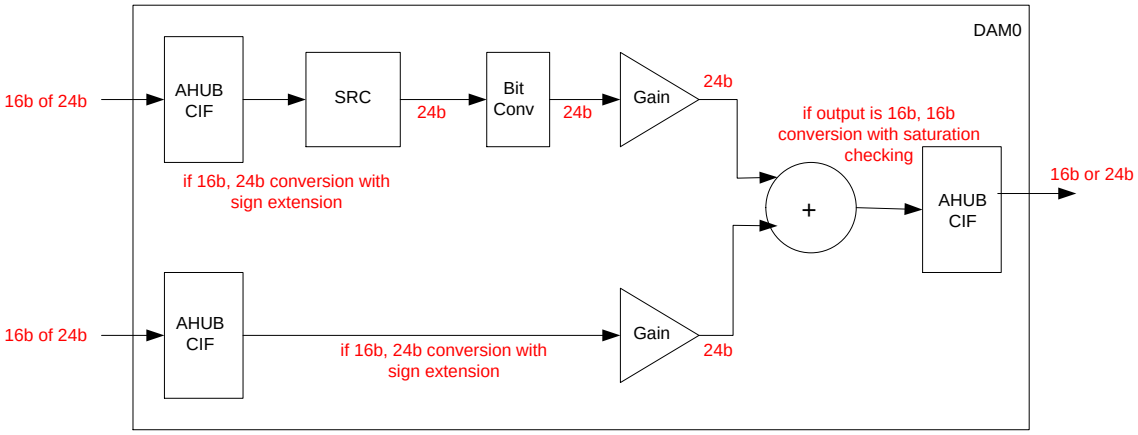


Multiplier Precision

While the input data and the step constants are 16 bit wide, the internal data is kept in the 24 bit format. For the multiplication, 24x16 multiplier is implemented. The width of accumulator is 43 bits for 40 bit + 3 guard bits for accumulation. Note that all data is signed.

While 16 bit samples are the only input format specified by P0, SRC can handle 24 bit input samples since the internal structure supports 24 bit samples. If RXCIF gets 16 bit input samples, it passes the samples untouched to the DAM core logic. The DAM core logic converts upgrades their precision to 24 bits with sign extension automatically before those samples are used in computation. If RXCIF gets 24 bits input samples, it can still forward those samples untouched to DAM. DAM will identify the bit width of the samples and use those as is without sign extension.

The reverse conversion should happen in the output stage of TXCIF. The internal 24 bit samples are converted to 16 bits with saturation checking before they are passed to TXCIF by the signal merger. If the output format is 24 bits, the samples can be forwarded to TXCIF untouched.



22.3.1.3 Channel Conversion

LLF SRC only supports 16 bit input mono streams.

22.3.1.4 Gain Controller

The gain controller supports a 16-bit volume level; and performs a 24x16 multiply with 24 bit output. While the LLF path only supports 16 bit data, the HQF will generate 24 bit output. To reuse the gain controller both in the LLF and the HQF paths, the input and output bit width is fixed to 24 bits. Also, the gain controller should support multi-channel stream, up to 8. Each gain controller has its own gain value.

The gain value is specified in the Q12 unsigned fractional format. In the binary representation,

iiii.ffffffffffff

where *i* and *f* represent integer and fractional binary numbers, respectively.

With the 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 bit to generate normalized output.

22.3.1.5 Mixer

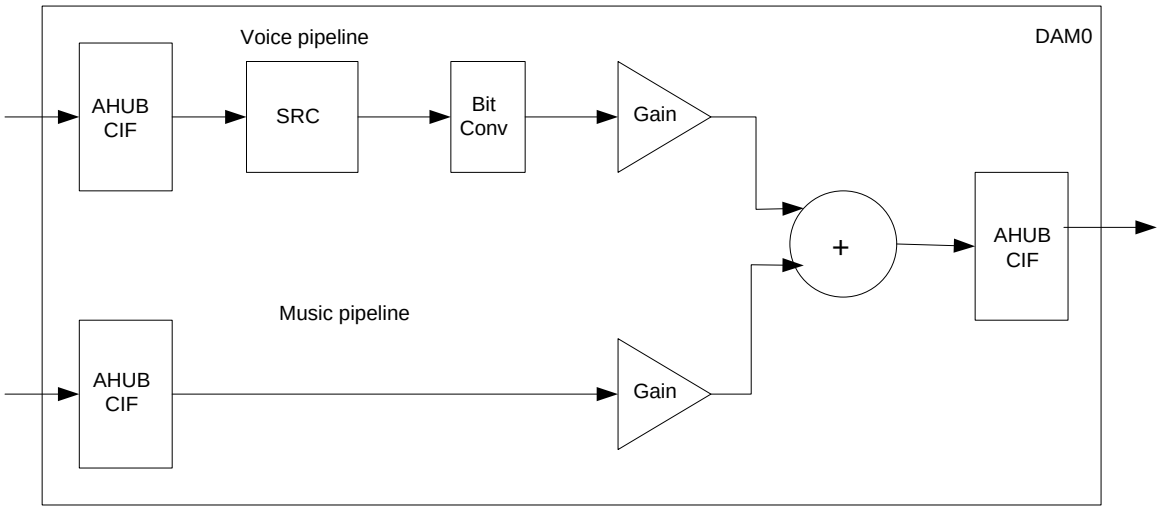
The mixer is simply a 24 bit adder. The output data from the gain controllers is added to generate a 24 bit value, which is sent to AHUBCIF TX. AHUBCIF converts the stream into user-specified bit format.

When the mixer tries to pick up the data from each channel, it is possible that one channel has data available, while the other channel doesn't. While this data synchronization can be specified in the configuration register, it is recommended to use a certain values for the field of each channel. Mostly, the data for the LLF channel is a stream input, while the data for the second channel comes from memory via DMA. In this scenario, the data in the second channel fills the channel pipeline, FIFOs, while the pipeline of the first channel is empty. Normally, the sample in the second channel should wait for the first channel for mixing, but the sample in the first channel should not wait for the second channel because the latency is critical in the first channel. The synchronization between the samples can be controlled by the configuration register settings.

When the mixer merges samples from multiple channels, the sum can get out of precision limit due to inappropriate gain values in the gain controller. Due to the bit with limitation, those out-of-limit samples will be clipped to the maximum values that the precision allows. If this happens frequently, the sound quality will be greatly degraded. To avoid this kind of situation, DAM notifies CPU to adjust the gain values through keeping the number clipped samples and an interrupt.

22.3.1.6 Error handling

While DAM expects constant data streams from CIFs, it may encounter some erroneous situations such as FIFO empty or FIFO full in its pipelines. Those can happen due to external issues somewhere in the system. A good example is APB DMA runs out of data when a system bus is congested so that CPU cannot access the DMA for the next batch of data for transmission. The FIFOs in DAM will be depleted, but it should still generate output stream.



The diagram shows a conversion case where a voice stream and a HIFI stream are mixed in DAM. SRC is the only one that can change the sampling frequency and all other module just obey the incoming sampling rate. There are some corner cases that should be carefully described.

- Whenever any sample is ready in AHUBCIF TX, it should send it to AHUB. If AHUBCIF TX doesn't get the ready signal when it has a sample to send, it will backpressure the rest of the pipeline.
- The mixer unit merges two samples from both pipelines. If both samples are ready, it mixes them and send it to AHUBCIF. If a sample from the the voice pipeline is ready, while the music pipeline is not, the mixer takes NULL data for the music pipeline and mixes it with the voice sample. If a sample from the music pipeline is ready, while the voice sample is not, it should wait for a sample from the voice pipeline. This only applies when both pipeline is turned on. If only one pipeline is turned on, the mixer sends out a sample whenever it is ready from the previous stage. The synchronization behavior is controlled by the configuration register.
- Depending on the configuration register programming, SRC generates an output sample stream from an input sample stream with different frequencies. If it gets backpressure from the next stage, it just stop generating samples until the condition changes. Also, if the input stream is not ready for SRC, it will just wait for the next sample.

If AHUBCIF RX gets backpressure from the next stage, it'll just hold the incoming AHUB bus until the condition changes.

22.4 I2S

The I^2S Controller has been designed to transfer streaming audio-data between the system memory and an audio-codec (audio-codec is an analog-to-digital (A/D) and/or digital-to-analog (D/A) converter). 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 (a simple serial link specifically designed for digital audio).

The I^2S controller supports point-to-point serial interface for the I^2S digital audio streams. I^2S-compatible products, such as compact disc players, digital audio tape devices, digital sound processors, and those with digital TV sound may be directly connected to the I2S controller. Controller also supports the PCM, telephony (network) and TDM 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 will be 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 these modes are synced up by a frame-sync signal (same as left-right-channel-clock –LRCK—in case of normal I2S communication).

I2S can be hooked up to any audio modules through AHUB. If I2S needs to access the system memory, it can transmit/receive transactions to/from APBIF, which can trigger the APB-bridge DMA to transfer data to and from its FIFOs. Controller supports two-FIFOs, FIFO-1 and FIFO-2 which can operate bi-directional in normal I2S modes but only uni-directional in PCM/NW/TDM modes.

Features

- Philips inter-IC-sound (I^2S) bus specification compliant
- PCM and NW mode support
- Can operate in both Master and Slave modes
- Supports I^2S, 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.
- Two FIFOs and control logic. In I2S basic modes, both the FIFOs can be configured for Tx or Rx. In PCM/NW mode, FIFO-1 will act 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 NW-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 can be any number of clocks in slave-mode.
- Supports TDM mode with programmability in number of slots, active slots and slot size.
- Same LRCK/FSYNC for both Transmission and reception of data.
- 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 device clock supported is 48 MHz
- The number of slots can be programmed from 1 to maximum 8 in TDM mode.

The I^2S controller supports bidirectional audio streams. It can operate in half-duplex or full-duplex 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 multiple of 4 starts from 8 bit.

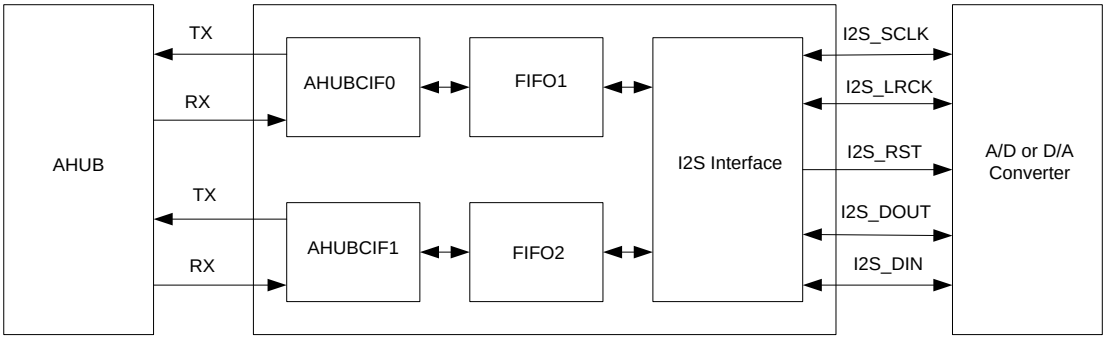
In Master mode, the I^2S controller signals the synchronization of all I^2S data transactions. The I^2S controller module generates the SCLK bit clock (I2S_SCLK,) and Left-Right channel clock LRCK (I2S_LRC) to the A/D or D/A converter. The A/D or D/A converter synchronizes the input or output audio data 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.

I^2S has two AHUBCIFs, which can be configured to either RX or TX depending on the direction of FIFOx and DOUT/DIN.

Note: Legacy naming of FIFO1 and FIFO2 are still used in the figure below, while AHUBCIFs are numbered from 0 to 1.

Figure 43. I2S Functional Block Diagram



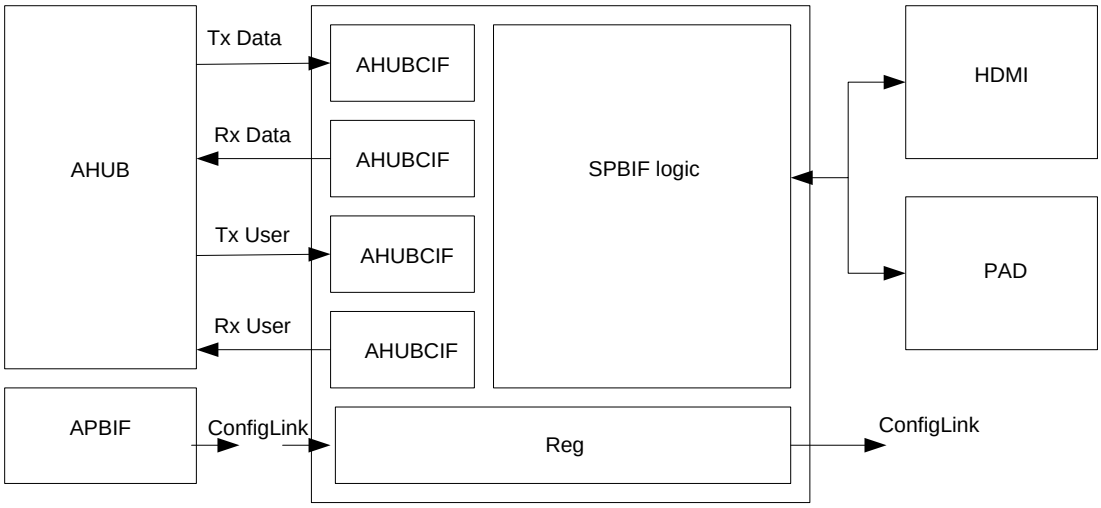
The configuration registers are programmed through a serial link from APBIF, ConfigLink. The interface has 1 bit data line for input and output and runs under the APB clock domain.

In order to avoid empty/full FIFO, a flow control mechanism is necessary in I2S. In the TX FIFO, an empty FIFO can cause sample dropping in the output stream. I2S should either add samples as specified by the configuration register or try to avoid the situation from the beginning. Also, if the RX FIFO is full in I2S, it should either drop some samples until there is room for new samples in the FIFO or try to avoid the situation with flow controlling.

22.5 SPDIF

The SPDIF interface is intended to support both professional and consumer applications, and therefore is a complete IEC-958 standard interface. When used in a professional application, the interface is primarily intended to carry monophonic or stereophonic programs, at a 48 kHz sampling frequency and with a resolution of up to 24 bits per sample; it may alternatively be used to carry signals sampled at 32 kHz or 44.1 kHz. When used in a consumer application, the interface is primarily intended to carry stereophonic programs, with a resolution of up to 20 bits per sample, an extension to 24 bits per sample being possible. When used for other purposes, the interface is primarily intended to carry audio data coded other than as linear PCM coded audio samples. Provision is 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.

SPDIF receives and/or transits data through AHUB from/to other AHUB clients. The register programming is performed through the ConfigLink interface, which is a daisy chain of a serial data line.



Features

- Support five data format.
  - 16-bit.
  - 20-bit.
  - 24-bit.
  - Raw.
  - 16-bit packed.
- Support “autolock” mode to automatically detect the “spdifin” sample rate and lock onto the data stream.
- Support “override” mode to provide a manual control to sample the “spdifin” data stream.
- Provide a loopback mode to route the “spdifout” back to “spdifin” for self-testing.
- SPDIFOUT (transmitter).
  - 16-word data FIFO for storage of outgoing audio data. (AHUBCIF)
  - 4-word user FIFO for storage of outgoing user data. (AHUBCIF)
  - 6-word page buffer for storage of outgoing channel status. (Config Reg)
- SPDIFIN (receiver).
  - 16-word data FIFO for storage of incoming audio data. (AHUBCIF)
  - 4-word user FIFO for storage of incoming user data. (AHUBCIF)
- 6-word page buffer for storage of incoming channel status. (Config Reg)

22.5.1 SPDIF Functionality

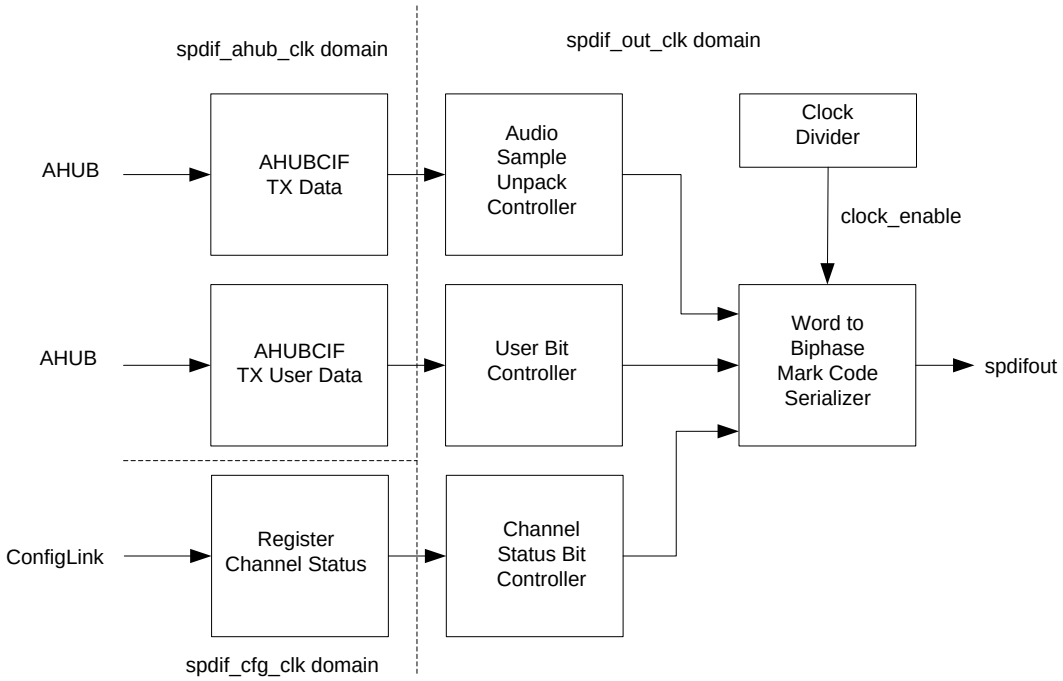
22.5.1.1 SPDIFOUT

The SPDIFOUT 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 44. 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.

Audio sample unpack controller

Based on the **PACK** register field, this controller will either break-up the packed data from the data FIFO or just let the data FIFO's data flow through unmodified to the "word to biphase mark code serializer" module. Also, this controller handles 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 will not be used.

User bit controller

When in 16, 20, 24-bit, or pack mode (**PACK** = 1 or **BIT.MODE** ≠ 11) and user data transmit is enabled (**TXU.E** = 1), this controller will temporary store an entry out of the user FIFO and feed user data, one bit **per subframe**, to the "word to biphase mark code serializer" module. Of course, this controller handles when to pop an entry out of the user FIFO. If user data transmit is disabled (**TXU.E** = 0), the user bit will be force 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 force to 0.

Word to Biphase Mark Code Serializer

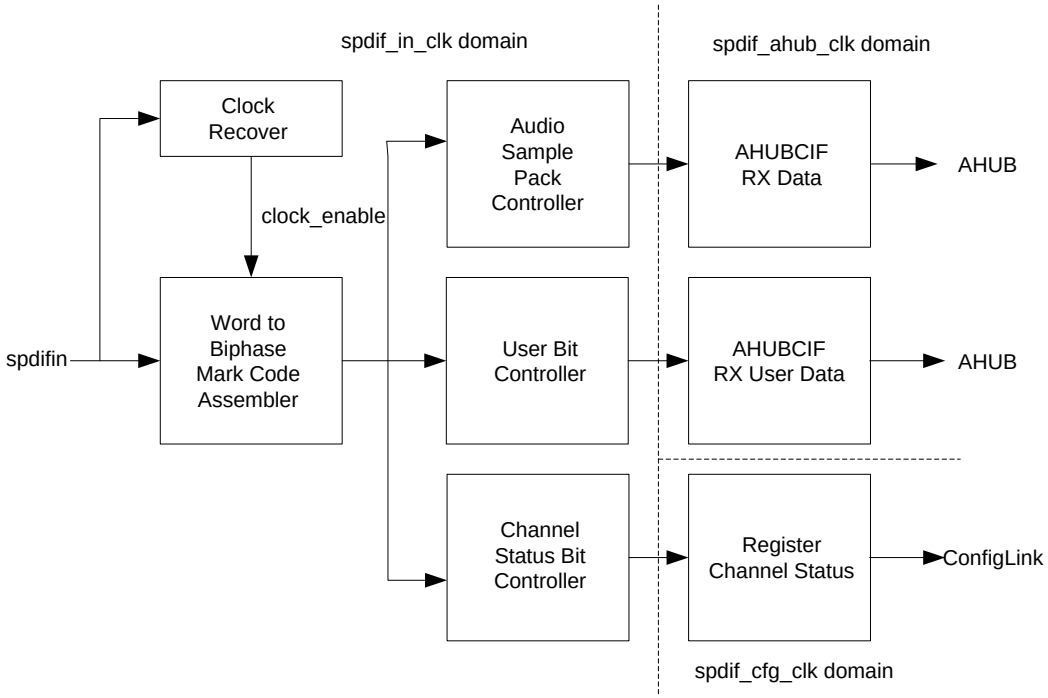
Based on the **PACK** and **BIT.MODE** register fields, this controller assembles a 32-bit word from the “Audio sample unpack controller” module, and/or “User bit controller” module, and/or “channel status bit controller” module. This 32-bit word will then be converted to the IEC 60958-3 biphase-mark code format and send out through the ‘spdifout’ signal. This controller also provides various status signals (i.e. end of transmitting a subframe, etc) back to the “unpack”, “user”, and “channel status” controllers.

22.5.1.2 SPDIFIN

The SPDIFIN module contains the following sub-blocks:

- A biphase 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 45. SPDIFIN Block Diagram



Pulse Counter

The purpose of this module is to perform the following three tasks, (1) count the number of “spdifin_clk” within the “spdifin” pulse, (2) generate a pulse to indicate that “spdifin” has toggled, and (3) indicate “spdifin” input is inactive.

Clk recover algorithm

Based on the inputs from the ‘Pulse counter’, this module calculate both, (1) the “spdifin” biphase period, and (2) the mid-point within the biphase period, in term of the number of “spdifin_clk”; the algorithm used is based on the 3 biphase period and 1 biphase period wide pulse in the “spdifin” data stream’s preamble phase. If the “override” mode is selected instead, the number of “spdifin_clk” within the biphase period and mid-point within the biphase period will be controlled by the **BIPHASE_DAT_STRB** and **BIPHASE_CLK_PERIOD** configuration register fields instead.

Clock enable decode (pre- and post-)

This module, which is used twice, contains the following functions, (1) to generate a “clk_enable” pulse in the middle of a “spdifin” biphase period for other modules to have a safe point for storing data from the “spdifin” data stream, and (2) to keep track of the bit position within a SPDIF subframe for the current “clk_enable” pulse. The reason why this module is used twice is because the “pre-“ module is used to search or re-lock to the “spdifin” data stream if the logic ever lost lock to the data stream. Once acquired lock for the first time, the “post-“ module will continue to use the previous saved biphase period and mid-point clock count to generate “clk_enable” even though the “pre-“ logic lost lock to the data stream and is in the process of re-locking (re-establishing the new biphase period and mid-point clock count) to the “spdifin” data stream.

Preamble decode (pre- and post-)

Based on the inputs from the ‘Clock enable decode’, this module provides miscellaneous decode such as, (1) when a preamble is detected from the “spdifin” data stream, (2) the beginning of a block, (3) whether the current subframe is channel A or channel B, and (4) when is the end of a subframe, etc. to other modules. This module is also used twice, one for each of the ‘Clock enable decode’ modules.

Lock checking

The purpose of this module is to act as a watcher/checker to the entire clock recovery logic. It determines, (1) when a lock to the “spdifin” data stream is established, (2) when a lock to the “spdifin” data stream is lost so that a re-lock is needed, (3) when to update the new biphase period and mid-point clock count values within both of the ‘clock enable decode’ modules.

22.5.1.3 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 convert/assemble the bit data into a 32-bit word. This 32-bit word will then be read by “audio sample pack controller”, “user bit controller”, and “channel status bit controller” modules for further processing. This controller also provides various status signals (i.e. end of receiving a subframe, etc) to the “pack”, “user”, and “channel status” controller.

22.5.1.4 Audio sample pack controller

Based on the **PACK** register field, this controller will either combine two audio data or just let the audio data flow through unmodified from the “biphase mark code to word assembler” module to the data FIFO. Also, this controller handles when to push an entry into the data FIFO.

22.5.1.5 Data FIFO

This 16-word deep FIFO is used to store audio samples written by the “audio sample pack controller” module; in 16, 20, 24-bit, or pack mode, it also stores non-audio data such as parity, channel status, user, validity, and preamble bits. Furthermore, this FIFO serves as a bridge between the ‘spdif_in_clk’ and ‘spdif_ahub_clk’ domain.

22.5.1.6 User bit controller

This controller will save the user bit from each subframe and assemble them into a 32-bit word before pushing an entry into the user FIFO.

22.5.1.7 User FIFO

This 4-word deep FIFO is used to store user data written by the “user bit controller” module. Also, this FIFO serves as a bridge between the ‘spdif_in_clk’ and ‘spdif_ahub_clk’ domain.

22.5.1.8 Channel status bit controller

This controller handles saving the channel status bit from channel A (subframe 1) **only** and assemble them into a 32-bit word before writing into the 6-word channel status page buffer which is also resided within this controller.

22.5.1.9 Flow Controller

The flow controller is implemented in both I2S and SPDIF to match each other’s output. A common sub-module should be implemented and instantiated as needed.

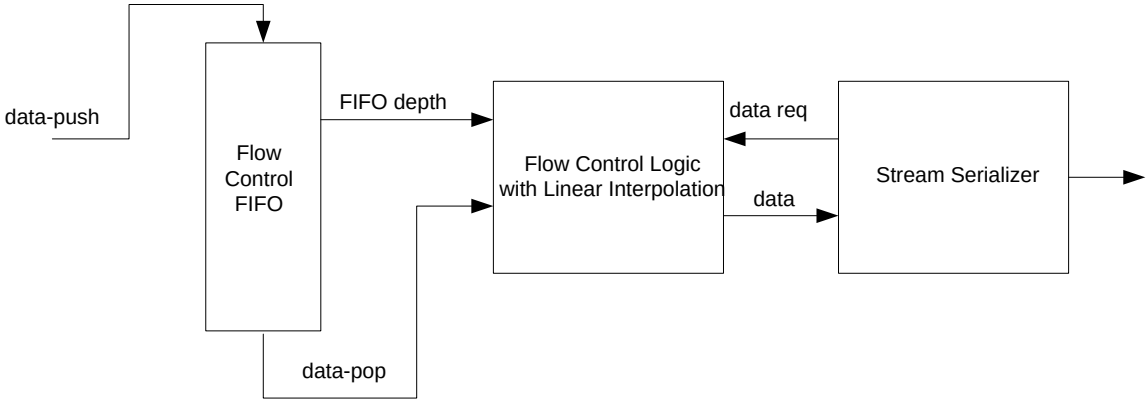
When communicating with a baseband processor and a CODEC, their operating clock frequencies can be a little bit different and not accurately synchronized to each other even their nominal frequencies are the same. Also, when sampling rate conversion is performed, the converted sampling rate frequency can be a little off from the SPDIF interface clock rate. For example, the sample rate conversion between 8 kHz and 44.1 kHz can leads to either data skipping or blanking, because 44.1 kHz output cannot be exactly aligned with 8 kHz input. Even the rate conversion is between 8 kHz and 48 kHz, their actual frequencies may not be ratio-synchronized because their clock sources can be different. If the error from the clock difference accumulates, the audio quality eventually suffers from skipping or blanking.

There are two exceptional events that SPDIF should pay special attention to, FIFO empty in the SPDIF TX and FIFO full in the SPDIF RX. The other cases, FIFO full in the SPDIF TX and FIFO empty in the SPDIF RX are not the critical events and the AHUB subsystem can minimize the impact through back-pressuring the pipeline or simply idling.

These events can happen in two different types of operations, DMA operations and stream operations. With the DMA operations, the exceptions rarely happens because SW programs DMA properly to avoid those and the DMA transactions are much faster than the external SPDIF transactions. However, with the stream transactions, there are good chances of those exceptions happening due to the asynchronous nature of the AHUB subsystem and the external SPDIF interface. Therefore, two different types of flow control mechanism are proposed in SPDIF to handle them appropriately.

The first one is a simple replication and drop. With the SPDIF RX, if FIFO is full, next incoming samples are simply dropped until FIFO has some room for new samples. With the SPDIF TX, if FIFO is empty, SPDIF can either replicate the last known sample or send out just 0’s depending on the register configuration. When either of these exceptions happen, SPDIF should log the situation and trigger an corresponding interrupt, if un-masked, to the master as soon as possible. This scheme usually can be used when SPDIF is pipelined with DMA.

The other scheme is to implement a linear interpolation filter to smooth out the samples in case of empty or full FIFO. This is used in SPDIF TX only, but not in SPDIF RX. SW specifies a pre-determined threshold in FIFO to control flow. If the FIFO depth is greater than the threshold, SPDIF increases the step size to reduce the number of sample per time unit, which will decrease the FIFO depth gradually. When the FIFO depth becomes the threshold value, the step size is back to normal. If the FIFO depth is less than the threshold, SPDIF decreases the step size to produce more samples per time unit, which will increase the FIFO depth gradually. This scheme is useful for the use case where a voice stream comes in an SPDIF interface and the processed stream go out from other I2S interfaces. A block diagram is shown below.



If the FIFO status is normal, the interpolation filter uses the regular step size, which will generates the identical output sample. If the input rate is faster than the output rate, the more symbols will be in the FIFO, which can cause eventual FIFO overflow. To avoid overflow, SPDIF takes a bigger step size to generate less output symbols per unit time. The range of the step sizes for each clock frequency is SW programmable. If the input rate is slow than the output rate, FIFO will underflow eventually. In that case, a smaller step is used to generate more output symbols per unit time. While we assume the worst clock difference is 100 ppm (0.01%), the step size variance is programmable for the optimal adjustment.

Normally, SPDIF takes the standard step size *S* to process input audio signals. If the FIFO level becomes greater than the pre-defined (programmable) threshold level, SRC replaces step size *S* with $S + \Delta$ to control the sample rate conversion flow. When SPDIF eventually catches up the input rate and the FIFO levels goes down to the level, it replaces back step size $S + \Delta$ with *S*. To avoid underflow, SPDIF takes $S - \Delta$. The threshold level and the various step sizes are SW programmable. Since both filters need to execute several 24x16 multiplication, a 24x16 multiply-accumulate (MAC) is used and can be shared, if possible. Note that both input to and output can be multi-channel PCMs.

In order to avoid accumulation of error through adding step size, a register programmable can be used to reset the current time every N samples. After SPDIF takes N input samples, it should reset the current time value and send out the input sample intact. If the FIFO depth is not at the threshold level, the adjusted step size is applied again from the reset state.

22.5.2 SPDIF Programming Guidelines

22.5.2.1 16-, 20-, 24-bit Mode

During transmission, the audio data is taken from the TXdata FIFO, the channel status bit is taken from the TXchannel 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.

22.5.2.2 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:

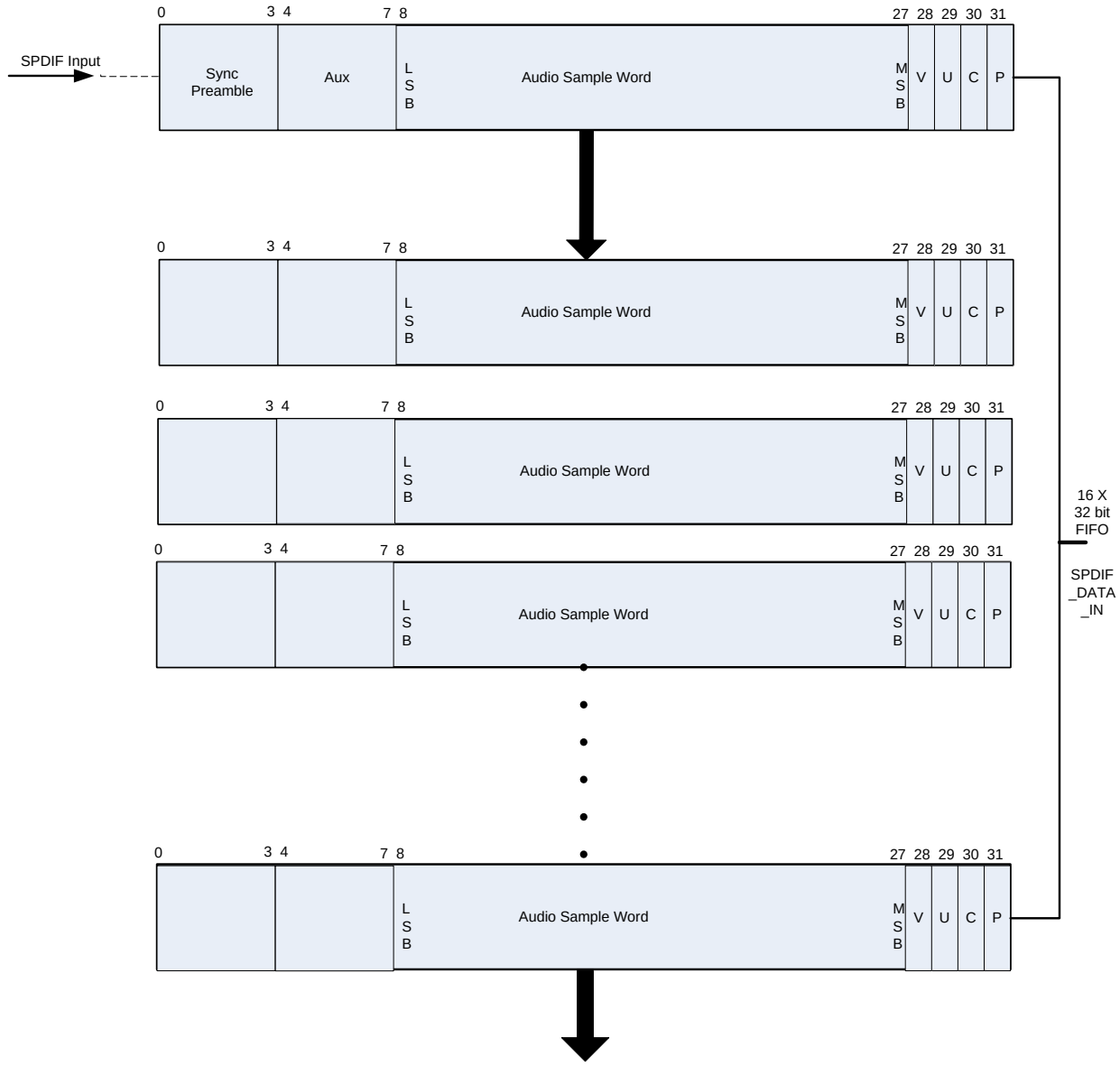
Preceding State in the Parity Bit: 0/1

Figure 46. 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 47. S/PDIF Data Format for Raw Mode



22.5.2.3 Pack Mode

During transmission, the audio data is taken from TX data FIFO which contains two 16-bit data (the upper 16-bit is for channel B or sub-frame 2, the lower 16-bit is for channel A or sub-frame 1, the channel status bits are taken from TX channel status page buffer and the user status bits are from 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 which contains two 16-bit data (the upper 16-bit being channel B data and the lower 16-bit being channel A data), the channel status bits are stored in the RX channel status page buffer and the user status bits are stored in the RX user FIFO. The reception of the channel bits start after the B-preamble is detected.

22.5.2.4 Module Reset

The application may initialize or reset the S/P-DIF module via the clock and reset (CAR) control's RST_DEVICES_L_0 register.

22.5.2.5 Clock Source/Divider Control

The SPDIFOUT clock source/divider can be selected by programming CAR's CLK_SOURCE_SPDIF register. Refer to the SPDIF 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 over-sample frequency.

22.5.2.6 Clock Enable Control

The application may enable the S/P-DIF clocks by programming CAR's CLK_OUT_ENB_L_0 register.

22.5.2.7 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 over-sampling technique

In addition to reading the sampling rate from the channel status information, one can also read the PERIOD field in register STROBE_CTRL_0 to estimate the incoming sample rate. The last two digits of PERIOD indicates 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 sub frame 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. As time goes by, the strobe point will vary less and less because the hardware has already adjusted to all the variations seen in the data stream.

22.5.2.8 S/P-DIF 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 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.

22.5.2.9 S/P-DIF Receive Data Flow

To enable the reception of RX_FIFO set (RX_EN field of 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 DATA_FIFO_CSR register) if the Rx interrupt enable bit is set.

22.5.2.10 Method of Filling/Retrieve Data from TX/RX User/Data FIFO

Use DMA method by programming the APBDMA 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.

22.6 Audio Hub 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 3 we only support replicating for mono and stereo frames due to the FIFO size limitation. If it's more than 2 channels, the samples in FIFO is used to fill other muti-channels.

DISABLE -> Disable replication. All zero's 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 channels 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 is received.

i : start transfer when i words is received.

For example, if FIFO_THRESHOLD is set to 3, then RxCif gives rd_req to 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 SPDIF. For all other cases default value of zero is the appropriate setting.

22.6.1 APBIF Registers

22.6.1.1 APBIF_CHANNEL0_CTRL_0

Offset: 000h | Read/Write: R/W | Reset: 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

22.6.1.2 APBIF_CHANNEL0_CLEAR_0

Offset: 004h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
31	0x0	TX_SOFT_RESET: 0 = DISABLE 1 = ENABLE
30	0x0	RX_SOFT_RESET: 0 = DISABLE 1 = ENABLE

22.6.1.3 APBIF_CHANNEL0_STATUS_0

Offset: 008h | Read/Write: RO | Reset: 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

22.6.1.4 APBIF_CHANNEL0_TXFIFO_0

Offset: 00ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DATA

22.6.1.5 APBIF_CHANNEL0_RXFIFO_0

Offset: 010h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DATA

22.6.1.6 APBIF_AUDIOCIF_TX0_CTRL_0

Offset: 014h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS:

Bit	R/W	Reset	Description
			0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

22.6.1.7 APBIF_AUDIOCIF_RX0_CTRL_0

Offset: 018h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3

Bit	R/W	Reset	Description
			3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

22.6.1.8 APBIF_CHANNEL1_CTRL_0

Offset: 020h | Read/Write: R/W | Reset: 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

22.6.1.9 APBIF_CHANNEL1_CLEAR_0

Offset: 024h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
31	0x0	TX_SOFT_RESET: 0 = DISABLE 1 = ENABLE
30	0x0	RX_SOFT_RESET: 0 = DISABLE 1 = ENABLE

22.6.1.10 APBIF_CHANNEL1_STATUS_0

Offset: 028h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	TX_FREE_COUNT
23:16	X	RX_FREE_COUNT
1	X	TX_TRIG: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	X	RX_TRIG: 0 = DISABLE 1 = ENABLE

22.6.1.11 APBIF_CHANNEL1_TXFIFO_0

Offset: 02ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DATA

22.6.1.12 APBIF_CHANNEL1_RXFIFO_0

Offset: 030h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DATA

22.6.1.13 APBIF_AUDIOCIF_TX1_CTRL_0

Offset: 034h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

22.6.1.14 APBIF_AUDIOCIF_RX1_CTRL_0

Offset: 038h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8

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

22.6.1.15 APBIF_CHANNEL2_CTRL_0

Offset: 040h | Read/Write: R/W | Reset: 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

Bit	Reset	Description
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

22.6.1.16 APBIF_CHANNEL2_CLEAR_0

Offset: 044h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
31	0x0	TX_SOFT_RESET: 0 = DISABLE 1 = ENABLE
30	0x0	RX_SOFT_RESET: 0 = DISABLE 1 = ENABLE

22.6.1.17 APBIF_CHANNEL2_STATUS_0

Offset: 048h | Read/Write: RO | Reset: 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

22.6.1.18 APBIF_CHANNEL2_TXFIFO_0

Offset: 04ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DATA

22.6.1.19 APBIF_CHANNEL2_RXFIFO_0

Offset: 050h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DATA

22.6.1.20 APBIF_AUDIOCIF_TX2_CTRL_0

Offset: 054h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

22.6.1.21 APBIF_AUDIOCIF_RX2_CTRL_0

Offset: 058h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

22.6.1.22 APBIF_CHANNEL3_CTRL_0

Offset: 060h | Read/Write: R/W | Reset: 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

22.6.1.23 APBIF_CHANNEL3_CLEAR_0

Offset: 064h | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
31	0x0	TX_SOFT_RESET: 0 = DISABLE 1 = ENABLE
30	0x0	RX_SOFT_RESET: 0 = DISABLE 1 = ENABLE

22.6.1.24 APBIF_CHANNEL3_STATUS_0

Offset: 068h | Read/Write: RO | Reset: 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

22.6.1.25 APBIF_CHANNEL3_TXFIFO_0

Offset: 06ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DATA

22.6.1.26 APBIF_CHANNEL3_RXFIFO_0

Offset: 070h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DATA

22.6.1.27 APBIF_AUDIOCIF_TX3_CTRL_0

Offset: 074h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	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
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

22.6.1.28 APBIF_AUDIOCIF_RX3_CTRL_0

Offset: 078h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	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
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

22.6.1.29 APBIF_CONFIG_LINK_CTRL_0

Offset: 080h | Read/Write: R/W | Reset: 0bxxxx10000000000010000000000x000

Bit	R/W	Reset	Description
31:28	RO	X	MASTER_FIFO_FULL_CNT: Config link master fifo full cnt
27:16	RW	0x800	TIMEOUT_CNT: if transaction is timed-out after waiting for these manycycles, interrupt will be generated in APBIF_INT_STATUS.CONFIG_LINK_TIMEOUT
15:4	RW	0x800	IDLE_CNT: Turn-off pclk to config_link slaves after the link is idle forthese many cycles
2	RW	0x0	CG_EN: 1 : clock gating is enabled. 0: free running. 0 = DISABLE 1 = ENABLE
1	RW	0x0	CLEAR_TIMEOUT_CNTR: Write 1 to clear time-out cntr.(Write-only) 0 = DISABLE 1 = ENABLE
0	RW	0x0	SOFT_RESET: Resets config_link master and slaves.(Write-only) 0 = DISABLE 1 = ENABLE

22.6.1.30 APBIF_MISC_CTRL_0

Offset: 084h | Read/Write: R/W | Reset: 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
4:0	RW	0x0	OBS_SEL: OBS Bus selection mux control

22.6.1.31 APBIF_APBDMA_LIVE_STATUS_0

Offset: 088h | Read/Write: RO | Reset: 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
26	X	CH1_TX_CIF_FIFO_FULL:

Bit	Reset	Description
		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

22.6.1.32 APBIF_I2S_LIVE_STATUS_0

Offset: 08ch | Read/Write: RO | Reset: 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

22.6.1.33 APBIF_DAM0_LIVE_STATUS_0

Offset: 090h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

22.6.1.34 APBIF_DAM1_LIVE_STATUS_0

Offset: 098h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

22.6.1.35 APBIF_DAM2_LIVE_STATUS_0

Offset: 0a0h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

22.6.1.36 APBIF_SPDIF_LIVE_STATUS_0

Offset: 0a8h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
19	X	SPDIF_TXU_BSY_RSVD: This field is redundant and is deprecated 0 = DISABLE 1 = ENABLE
18	X	SPDIF_TXC_BSY_RSVD: This field is redundant and is deprecated 0 = DISABLE 1 = ENABLE
17	X	SPDIF_TX_BSY_RSVD: This field is redundant and is deprecated 0 = DISABLE 1 = ENABLE
16	X	SPDIF_RX_BSY_RSVD: This field is redundant and is deprecated 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

Bit	Reset	Description
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

22.6.1.37 APBIF_I2S_INT_MASK_0

Since 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: 0b0h | Read/Write: R/W | Reset: 0b111111111111111xxxxxx1111111111

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

Bit	Reset	Description
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
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

22.6.1.38 APBIF_DAM_INT_MASK_0

Offset: 0b4h | Read/Write: R/W | Reset: 0b1xx1xx111xx1xx111xx1xx11

Bit	Reset	Description
23	0x1	DAM2_CLIP_INTR: 0 = DISABLE 1 = ENABLE
20	0x1	DAM2_TX_DONE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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
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

22.6.1.39 APBIF_SPDIF_INT_MASK_0

Offset: 0bch | Read/Write: R/W | Reset: 0b1xxx111111111111

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

Bit	Reset	Description
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
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

22.6.1.40 APBIF_APBIF_INT_MASK_0

Offset: 0c0h | Read/Write: R/W | Reset: 0b11111111111111111

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

Bit	Reset	Description
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
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

22.6.1.41 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: 0c8h | Read/Write: R/W | Reset: 0b00000000000000xxxxxx0000000000

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

Bit	Reset	Description
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
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

Bit	Reset	Description
0	0x0	I2S0_TX_DONE: 0 = DISABLE 1 = ENABLE

22.6.1.42 APBIF_DAM_INT_STATUS_0

Offset: 0cch | Read/Write: R/W | Reset: 0b0xx0xx000xx0xx000xx0xx00

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

22.6.1.43 APBIF_SPDIF_INT_STATUS_0

Offset: 0d4h | Read/Write: R/W | Reset: 0b0xxx000000000000

Bit	Reset	Description
15	0x0	SPDIF_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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
1	0x0	SPDIF_DATA_TX_DONE: 0 = DISABLE 1 = ENABLE
0	0x0	SPDIF_DATA_RX_DONE: 0 = DISABLE 1 = ENABLE

22.6.1.44 APBIF_APBIF_INT_STATUS_0

Offset: 0d8h | Read/Write: R/W | Reset: 0b00000000000000000

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

Bit	Reset	Description
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
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

22.6.1.45 APBIF_I2S_INT_SOURCE_0

Interrupt Source

Offset: 0e0h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	I2S4_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
29	X	I2S3_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	X	I2S2_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
27	X	I2S1_FLOW_CTL_INT: 0 = DISABLE 1 = ENABLE
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

Bit	Reset	Description
4	X	I2S2_TX_DONE: 0 = DISABLE 1 = ENABLE
3	X	I2S1_RX_DONE: 0 = DISABLE 1 = ENABLE
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

22.6.1.46 APBIF_DAM_INT_SOURCE_0

Offset: 0e4h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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

Bit	Reset	Description
0	X	DAM0_RX0_DONE: 0 = DISABLE 1 = ENABLE

22.6.1.47 APBIF_SPDIF_INT_SOURCE_0

Offset: 0ech | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

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

22.6.1.48 APBIF_APBIF_INT_SOURCE_0

Offset: 0f0h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxx

Bit	Reset	Description
16	X	CONFIG_LINK_TIMEOUT: 0 = DISABLE 1 = ENABLE
15	X	APBIF_CH3_APBDMA_RX_OVERRUN: 0 = DISABLE 1 = ENABLE
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

22.6.1.49 APBIF_I2S_INT_SET_0

Interrupt set

Offset: 0f8h | Read/Write: R/W | Reset: 0b0000000000000000xxxxxx0000000000

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

22.6.1.50 APBIF_DAM_INT_SET_0

Offset: 0fch | Read/Write: R/W | Reset: 0b0xx0xx000xx0xx000xx0xx00

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

22.6.1.51 APBIF_SPDIF_INT_SET_0

Offset: 100h | Read/Write: R/W | Reset: 0b0xxx000000000000

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

22.6.1.52 APBIF_APBIF_INT_SET_0

Offset: 104h | Read/Write: R/W | Reset: 0b00000000000000000

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

Packet is Big Endian and HEADER is transmitted first. This packet is not used for Tegra 3.

22.6.2 Audio Registers

22.6.2.1 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 a 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: 000h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.2 AUDIO_APBIF_RX1_0

Offset: 004h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.3 AUDIO_APBIF_RX2_0

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.4 AUDIO_APBIF_RX3_0

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.5 AUDIO_I2S0_RX0_0

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.6 AUDIO_I2S1_RX0_0

Offset: 014h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.7 AUDIO_I2S2_RX0_0

Offset: 018h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.8 AUDIO_I2S3_RX0_0

Offset: 01ch | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.9 AUDIO_I2S4_RX0_0

Offset: 020h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.10 AUDIO_DAM0_RX0_0

Offset: 024h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.11 AUDIO_DAM0_RX1_0

Offset: 028h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.12 AUDIO_DAM1_RX0_0

Offset: 02ch | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.13 AUDIO_DAM1_RX1_0

Offset: 030h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.14 AUDIO_DAM2_RX0_0

Offset: 034h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.15 AUDIO_DAM2_RX1_0

Offset: 038h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.16 AUDIO_SPDIF_RX0_0

Offset: 03ch | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

22.6.2.17 AUDIO_SPDIF_RX1_0

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	SPDIF_TX1
12	0x0	SPDIF_TX0
11	0x0	DAM2_TX0
10	0x0	DAM1_TX0
9	0x0	DAM0_TX0
8	0x0	I2S4_TX0
7	0x0	I2S3_TX0
6	0x0	I2S2_TX0
5	0x0	I2S1_TX0
4	0x0	I2S0_TX0
3	0x0	APBIF_TX3
2	0x0	APBIF_TX2
1	0x0	APBIF_TX1
0	0x0	APBIF_TX0

INTERNAL PACKETS

- AUDIO_RX_REG(0, APBIF, apbif, 0, APBIF, apbif, 0)
- AUDIO_RX_REG(1, APBFI, apbif, 0, APBIF, apbif, 1)
- AUDIO_RX_REG(2, APBIF, apbif, 0, APBIF, apbif, 2)
- AUDIO_RX_REG(3, APBIF, apbif, 0, APBIF, apbif, 3)
- AUDIO_RX_REG(4, I2S, i2s, 0, I2S0, i2s0, 0)
- AUDIO_RX_REG(5, I2S, i2s, 1, I2S1, i2s1, 0)
- AUDIO_RX_REG(6, I2S, i2s, 2, I2S2, i2s2, 0)
- AUDIO_RX_REG(7, I2S, i2s, 3, I2S3, i2s3, 0)
- AUDIO_RX_REG(8, I2S, i2s, 4, I2S4, i2s4, 0)
- AUDIO_RX_REG(9, DAM, dam, 0, DAM0, dam0, 0)
- AUDIO_RX_REG(10, DAM, dam, 0, DAM0, dam0, 1)
- AUDIO_RX_REG(11, DAM, dam, 1, DAM1, dam1, 0)
- AUDIO_RX_REG(12, DAM, dam, 1, DAM1, dam1, 1)
- AUDIO_RX_REG(13, DAM, dam, 2, DAM2, dam2, 0)
- AUDIO_RX_REG(14, DAM, dam, 2, DAM2, dam2, 1)
- AUDIO_RX_REG(15, SPDIF, spdif, 0, SPDIF, spdif, 0)
- AUDIO_RX_REG(16, SPDIF, spdif, 0, SPDIF, spdif, 1)

22.6.3 DAM Registers

22.6.3.1 DAM_CTRL_0

Offset: 000h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	DISABLE	SOFT_RESET: This bit is Auto Cleared.Resets DAM logic including CIFs and only clears the DAM Enable & Channel Enables configuration bits SW 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
1	DISABLE	CG_EN: enable DAM second level clock gating 0 = DISABLE 1 = ENABLE
0	DISABLE	ENABLE: enable DAM 0 = DISABLE 1 = ENABLE

DAM_CLIP_0

Offset: 004h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RW	DISABLE	ENABLE: enable the counter 0 = DISABLE 1 = ENABLE
30:0	RO	X	COUNT: This value indicates the total actual number of succesive DAM Output Samples that are >= the CLIP_THRESHOLD_VALUE,This value can be greater than CLIP_THRESHOLD_COUNT

22.6.3.2 DAM_CLIP_THRESHOLD_0

Offset: 008h | Read/Write: R/W | Reset: 0b00000000011111111111111100000000

Bit	Reset	Description
31:8	0x7fff	VALUE: DAM output samples >= this value increments the CLIP Counter
7:0	0x0	COUNT: DAM Interrupt is generated when this number of consecutive samples has occured with magnitude >= to CLIP_THRESHOLD_VALUE. DAM generates Interrupt and moves on.Interrupt is again generated only after SW has cleared the CLIPE ENABLE & enable it again and on the next event happening

22.6.3.3 DAM_AUDIOCIF_OUT_CTRL_0

For the DAM output

DAM output is fixed to 24 bits. CIF should convert the output to the desired format

CLIENT_BITS : BIT24

Offset: 00ch | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8

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

22.6.3.4 DAM_CH0_CTRL_0

Session channel 0 -> LLF path

Conv	regular step size
8 -> 44.1	0x01738
8 -> 48	0x01555
16 -> 44.1	0x02e70
16 -> 48	0x02aaa
44.1-> 8	0x2c199
48 -> 8	0x30000
44.1-> 16	0x160cc
48 -> 16	0x18000

Offset: 010h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx0000xxx0

Bit	Reset	Description
31:16	X	STEP_RESET: Every a certain amount of the input samples, the current step value is reset to 0 to avoid error accumulation. 0 -> never reset The recommend values are listed below. 8 -> 44.1 STEP_RESET = 80, 800, 8000 8 -> 48 STEP_RESET = 1 16 -> 44.1 STEP_RESET = 160, 1600, 16000 16 -> 48 STEP_RESET = 1 44.1 -> 8 STEP_RESET = 441, 4410, 44100 44.1 -> 16 STEP_RESET = 441, 4410, 44100 48 -> 8 STEP_RESET = 0 48 -> 16 STEP_RESET = 0
11:8	X	FSIN: input sample rate 0 = FS8 1 = FS16 2 = FS44 3 = FS48
7:4	0x0	DATA_SYNC: When a sample is ready in this channel, the bit says the mixer should wait for data in other channels Each bit represent each channel, i.e. bit 1 -> CH 1. bit0 is meaningless for CH0 For CH0, 0b0000 is recommended since CH0 data should not wait for other channel.
0	DISABLE	ENABLE: 0 = DISABLE 1 = ENABLE

22.6.3.5 DAM_CH0_CONV_0

16 bit linear gain factor for both paths in Q12 unsigned fractional format

Offset: 014h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	GAIN: bit format: unsigned 4 bit fixed/12 bit fractional -> iiii.fffffffffff always positive number (no sign bit) 0x1000 is linear gain 1.0

22.6.3.6 DAM_AUDIOCIF_CH0_CTRL_0

DAM has 4 RX ports and 1 TX port to AUDIO for the P1 implementation. Only RX0 and RX1 is used for the P0 implementation for the channel0 input (LLF path)

Since LLF can handle only 16-bit mono data, CIF should be configured accordingly.

CLIENT_BITS : BIT16

CLIENT_CHANNELS : CH1

PACK : NOP

Offset: 01ch | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5

Bit	R/W	Reset	Description
			5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

22.6.3.7 DAM_CH1_CTRL_0

Session channel 1 -> HQF path in P1, no SRC in P0

Offset: 020h | Read/Write: R/W | Reset: 0b0001xxx0

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 represent each channel, i.e. bit 0 -> CH 0. bit1 is meaningless for CH1 For CH1, 0b0001 is recommended since CH1 data should wait for CH0 data.
0	DISABLE	ENABLE: 0 = DISABLE 1 = ENABLE

22.6.3.8 DAM_CH1_CONV_0

16-bit linear gain factor for both paths in Q12 unsigned fractional format.

Offset: 024h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	GAIN: bit format: unsigned 4 bit fixed/12-bit fractional -> iiii.ffffffffffff always positive number (no sign bit)

22.6.3.9 DAM_AUDIOCIF_CH1_CTRL_0

For the channel1 input

Offset: 02ch | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

22.6.4 I2S Registers

The I2S Controller is designed to transport streaming audio-data between the system memory and an audio-codec. The controller supports I2S format, Left Justified Mode format, Right Justified Mode format, and DSP mode format, as defined in the Philips inter-IC-sound (I2S) 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.

22.6.4.1 I2S_CTRL_0

This is I2S control register where we can 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: 000h | Read/Write: R/W |Reset: 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
26:24	0x0	OBS_SEL: OBS bus selection mux
14:12	0x0	FRAME_FORMAT: Frame format. 0: BASIC,LJM, RJM modes 1: DSP,PCM,NW,TDM modes 0 = LRCK_MODE 1 = FSYNC_MODE
10	0x0	MASTER: Controller Master/Slave mode selection. 0: Don't drive LRCK and clk 1: Drive LRCK and clk 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 Note: 4 bits per sample is NOT supported in Tegra 3. 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

22.6.4.2 I2S_TIMING_0

The CHANNEL_BIT_CNT field of this register is used to program the number of bit-clks per channel of LRCK (sampling rate) the I2S controller needs to send out in Master mode. This value is interpreted as follows:

a) DSP mode:- Number of bit clocks (sclk) within (LEFT +RIGHT) channel width.

b) Basic/LJM/RJM modes:- Number of bit clocks (sclk) within each individual channel width (LEFT or RIGHT).

c) 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-clk 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 the 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 relation:

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: 004h | Read/Write: R/W | Reset: 0b0x00000011111

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: No. of bit clocks in left or right channel. DSP mode: No. of bit clocks in LEFT+RIGHT channel. TDM/NW/PCM mode: No. of bit clocks in the frame

22.6.4.3 I2S_OFFSET_0

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000xxxxx00000000000

Bit	Reset	Description
26:16	0x0	RX_DATA_OFFSET: RX Data offset to fsync 0: No offset n: Data is offset by n bit clk wrt fsync
10:0	0x0	TX_DATA_OFFSET: TX Data offset to fsync 0: No offset n: Data is offset by n bit clk wrt fsync

22.6.4.4 I2S_CH_CTRL_0

Offset: 00ch | Read/Write: R/W | Reset: 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 bitclk cycle instead of driving it for full bit cloc

Bit	Reset	Description
		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
8	0x0	EDGE_CTRL: edge on which Data is driven 0: Drive data on negative edge and sample data on positive edge. 1: Drive data on positive edge and sample data on 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 bit 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

22.6.4.5 I2S_SLOT_CTRL_0

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
18:16	0x0	TOTAL_SLOTS: Number of slots per fsync 0 : frame has 1 slot 1 : frame has 2 slot n : frame has n+1 slot
15:8	0x0	RX_SLOT_ENABLES: Rx Slot enables. Used for TDM mode
7:0	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.

22.6.4.6 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 reg programming so that I2S can have either two RX or two RX AUDIO ports.

CIF RX port for I2S TX path

Offset: 014h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

22.6.4.7 I2S_AUDIOCIF_I2SRX_CTRL_0

CIF TX port for I2S RX path

Offset: 018h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

22.6.4.8 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 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.

FLOWCTRL

Offset: 01ch | Read/Write: R/W | Reset: 0b1

Bit	Reset	Description
31	QUAD	FILTER: to use linear or quadratic filter. Tegra 3 supports quadratic filter only. 0 = LINEAR 1 = QUAD

22.6.4.9 I2S_TX_STEP_0

Offset: 020h | Read/Write: R/W | Reset: 0b1000000000000000

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 greater than the clock difference too much. For example, the input frame sampling rate is 44.095kHz and the output frame sampling rate is 44.100kHz, 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 a recommended value. If the value is less than 3 in the above example the adjusted step size wont be able to catch up // the flow control FIFO will eventually become empty. This will lead to either NULL or replicating the last sample.

22.6.4.10 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, ifMONITOR_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 o the time step.

Flow Controller Monitor/Counter

Offset: 024h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxxxxxxxxxxxxxx0xx00

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

22.6.4.11 I2S_FLOW_TOTAL_0

Offset: 028h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	COUNTER: the number of total left channel samples processed

22.6.4.12 I2S_FLOW_OVER_0

Offset: 02ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	COUNTER: the number of +STEP_SIZE

22.6.4.13 I2S_FLOW_UNDER_0

Offset: 030h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	COUNTER: the number of -STEP_SIZE

22.6.4.14 I2S_LCOEF_1_4_0_0

The LCOEF_1_4 coefficients are {46, -1562, 8394, 28999, -3714, 480};

Offset: 034h | Read/Write: R/W | Reset: 0b0000000000101110

Bit	Reset	Description
15:0	0x2e	COEF: Coefficient 0

22.6.4.15 I2S_LCOEF_1_4_1_0

Offset: 038h | Read/Write: R/W | Reset: 0b1111100111100110

Bit	Reset	Description
15:0	0xf9e6	COEF: Coefficient 1

22.6.4.16 I2S_LCOEF_1_4_2_0

Offset: 03ch | Read/Write: R/W | Reset: 0b0010000011001010

Bit	Reset	Description
15:0	0x20ca	COEF: Coefficient 2

22.6.4.17 I2S_LCOEF_1_4_3_0

Offset: 040h | Read/Write: R/W | Reset: 0b0111000101000111

Bit	Reset	Description
15:0	0x7147	COEF: Coefficient 3

22.6.4.18 I2S_LCOEF_1_4_4_0

Offset: 044h | Read/Write: R/W | Reset: 0b1111000101111110

Bit	Reset	Description
15:0	0xf17e	COEF: Coefficient 4

22.6.4.19 I2S_LCOEF_1_4_5_0

Offset: 048h | Read/Write: R/W | Reset: 0b0000000111100000

Bit	Reset	Description
15:0	0x1e0	COEF: Coefficient 5

22.6.4.20 I2S_LCOEF_2_4_0_0

The LCOEF_2_4 coefficients are {279, -3477, 19463, 19463, -3477, 279}; Given the symmetry, we only use 3 registers.

Farrow Filter Fractional

Offset: 04ch | Read/Write: R/W | Reset: 0b0000000100010111

Bit	Reset	Description
15:0	0x117	COEF: Coefficient 0

22.6.4.21 I2S_LCOEF_2_4_1_0

Offset: 050h | Read/Write: R/W | Reset: 0b1111001001101011

Bit	Reset	Description
15:0	0xf26b	COEF: Coefficient 1

22.6.4.22 I2S_LCOEF_2_4_2_0

Offset: 054h | Read/Write: R/W | Reset: 0b0100110000000111

Bit	Reset	Description
15:0	0x4c07	COEF: Coefficient 2

22.6.5 SPDIF Registers

The SPDIF consists of the following two major modules:

(1) The SPDIFOUT sub-module, responsible for sending data to the "spdifout" port in IEC 60958-3 biphase-mark code format.

(2) The SPDIFIN sub-module, responsible for retrieving data to the "spdifin" port in IEC 60958-3 biphase-mark code format.

Clock rates required by each sub-controller for given sample rate:

Sample Rate (KHz)	SPDIFOUT clock freq (exact freq. MHz)	Period (ns)	SPDIFIN clock freq. (min. freq but <= 100MHz)
32.0	4.0960	244.141	16.3840 (48MHz typical)
44.1	5.6448	177.154	22.5790 (48MHz typical)
48.0	6.1440	162.760	24.5760 (48MHz typical)
88.2	11.2896	88.577	45.1584 (72MHz typical)
96.0	12.2880	81.380	49.1520 (72MHz typical)
176.4	22.5792	44.289	90.3168 (108MHz typical)
192.0	24.5760	40.690	98.3040 (108MHz typical)

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.

22.6.5.1 SPDIF_CTRL_0

Offset: 000h | Read/Write: R/W | Reset: 0b000000xx0000xxxx000000000

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

Bit	Reset	Description
27	0x0	TC_EN: Transmit Channel status: 1=enable, 0=disable. 0 = DISABLE 1 = ENABLE
26	0x0	TU_EN: Transmit user Data: 1=enable, 0=disable. 0 = DISABLE 1 = ENABLE
23	0x0	IE_P_RSVD: Deprecated.
22	0x0	IE_B_RSVD: Deprecated
21	0x0	IE_C_RSVD: Deprecated.
20	0x0	IE_U_RSVD: Deprecated.
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 bit needs to be padded to match the interface data bit size) 0 = DISABLE 1 = ENABLE
13:12	0x0	BIT_MODE: 00=16bit data 01=20bit data 10=24bit 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
10:8	0x0	OBS_SEL: OBS bus select control
7	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

22.6.5.2 SPDIF_STROBE_CTRL_0

SPDIF Data Strobe Control Register

Offset: 004h | Read/Write: R/W | Reset: 0bxxxxxxxx0xx00000xx000000

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 bi-phase clock period (in terms of the number of over-sampling clocks)
5:0	RW	0x0	CLOCK_PERIOD: Manual SPDIFIN bi-phase clock period (in terms of the number of over-sampling or 'spdifin' clocks)

22.6.5.3 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: 008h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

Bit	R/W	Reset	Description
1	RW	0x0	TRUNCATE: 0 = ROUND 1 = CHOP
0	RW	0x0	MONO_CONV: 0 = ZERO 1 = COPY

22.6.5.4 SPDIF_AUDIOCIF_RXDATA_CTRL_0

CIF TX port for SPDIF RX DATA

Offset: 00ch | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

22.6.5.5 SPDIF_AUDIOCIF_TXUSER_CTRL_0

CIF RX port for SPDIF USER data

Offset: 010h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
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

Bit	R/W	Reset	Description
			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

22.6.5.6 SPDIF_AUDIOCIF_RXUSER_CTRL_0

CIF TX port for SPDIF USER data

Offset: 014h | Read/Write: R/W | Reset: 0b0000x000xxxxx000x001x00100000x00

Bit	R/W	Reset	Description
31:28	RW	0x0	FIFO_THRESHOLD
26:24	RW	0x0	AUDIO_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
18:16	RW	0x0	CLIENT_CHANNELS: 0 = CH1 1 = CH2 2 = CH3 3 = CH4 4 = CH5 5 = CH6 6 = CH7 7 = CH8
14:12	RW	0x1	AUDIO_BITS: 0 = BIT4 1 = BIT8 2 = BIT12 3 = BIT16 4 = BIT20 5 = BIT24

Bit	R/W	Reset	Description
			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

22.6.5.7 SPDIF_CH_STA_RX_A_0

This 6-words 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 bit from channel A (subframe 1) will be saved into this page buffer.

SPDIF Channel Status Rx Page Buffer Register

Offset: 018h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	RX_C31_RX_C0: Channel status bits [31:0]; one bit per audio sample

22.6.5.8 SPDIF_CH_STA_RX_B_0

Offset: 01ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	RX_C63_RX_C32: Channel status bits [63:32]; one bit per audio sample

22.6.5.9 SPDIF_CH_STA_RX_C_0

Offset: 020h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	RX_C95_RX_C64: Channel status bits [95:64]; one bit per audio sample

22.6.5.10 SPDIF_CH_STA_RX_D_0

Offset: 024h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	RX_C127_RX_C96: Channel status bits [127:96]; one bit per audio sample

22.6.5.11 SPDIF_CH_STA_RX_E_0

Offset: 028h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	RX_C159_RX_C128: Channel status bits [159:128]; one bit per audio sample

22.6.5.12 SPDIF_CH_STA_RX_F_0

Offset: 02ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	RX_C191_RX_C160: Channel status bits [191:160]; one bit per audio sample

22.6.5.13 SPDIF_CH_STA_TX_A_0

This 6-words 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 send out to "both" subframes.

SPDIF Channel Status Tx Page Buffer Register

Offset: 030h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_C31_TX_C0: Channel status bits [31:0]; one bit per audio sample

22.6.5.14 SPDIF_CH_STA_TX_B_0

Offset: 034h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_C63_TX_C32: Channel status bits [63:32]; one bit per audio sample

22.6.5.15 SPDIF_CH_STA_TX_C_0

Offset: 038h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_C95_TX_C64: Channel status bits [95:64]; one bit per audio sample

22.6.5.16 SPDIF_CH_STA_TX_D_0

Offset: 03ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_C127_TX_C96: Channel status bits [127:96]; one bit per audio sample

22.6.5.17 SPDIF_CH_STA_TX_E_0

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_C159_TX_C128: Channel status bits [159:128]; one bit per audio sample

22.6.5.18 SPDIF_CH_STA_TX_F_0

Offset: 044h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_C191_TX_C160: Channel status bits [191:160]; one bit per audio sample

22.6.5.19 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: 070h | Read/Write: R/W | Reset: 0b1

Bit	Reset	Description
31	QUAD	FILTER: to use linear or quadratic filter. Tegra 3 supports quadratic filter only. 0 = LINEAR 1 = QUAD

22.6.5.20 SPDIF_TX_STEP_0

Offset: 074h | Read/Write: R/W | Reset: 0b1000000000000000

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 greater than the clock difference too much. For example, the input frame sampling rate is 44.095kHz and the output frame sampling rate is 44.100kHz, 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 a recommended value. If the value is less than 3 in the above example the adjusted step size won't be able to catch up // the flow control FIFO will eventually become empty. This will lead to either NULL or replicating the last sample.

22.6.5.21 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 o the time step.

Flow Controller Monitor/Counter

Offset: 078h | Read/Write: R/W | Reset: 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

22.6.5.22 SPDIF_FLOW_TOTAL_0

Offset: 07ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	COUNTER: the number of total left channel samples processed

22.6.5.23 SPDIF_FLOW_OVER_0

Offset: 080h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	COUNTER: the number of +STEP_SIZE

22.6.5.24 SPDIF_FLOW_UNDER_0

Offset: 084h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	COUNTER: the number of -STEP_SIZE

22.6.5.25 SPDIF_LCOEF_1_4_0_0

The LCOEF_1_4 coefficients are {46, -1562, 8394, 28999, -3714, 480};

Offset: 088h | Read/Write: R/W | Reset: 0b0000000000101110

Bit	Reset	Description
15:0	0x2e	COEF: Coefficient 0

22.6.5.26 SPDIF_LCOEF_1_4_1_0

Offset: 08ch | Read/Write: R/W | Reset: 0b1111100111100110

Bit	Reset	Description
15:0	0xf9e6	COEF: Coefficient 1

22.6.5.27 SPDIF_LCOEF_1_4_2_0

Offset: 090h | Read/Write: R/W | Reset: 0b0010000011001010

Bit	Reset	Description
15:0	0x20ca	COEF: Coefficient 2

22.6.5.28 SPDIF_LCOEF_1_4_3_0

Offset: 094h | Read/Write: R/W | Reset: 0b0111000101000111

Bit	Reset	Description
15:0	0x7147	COEF: Coefficient 3

22.6.5.29 SPDIF_LCOEF_1_4_4_0

Offset: 098h | Read/Write: R/W | Reset: 0b1111000101111110

Bit	Reset	Description
15:0	0xf17e	COEF: Coefficient 4

22.6.5.30 SPDIF_LCOEF_1_4_5_0

Offset: 09ch | Read/Write: R/W | Reset: 0b0000000111100000

Bit	Reset	Description
15:0	0x1e0	COEF: Coefficient 5

22.6.5.31 SPDIF_LCOEF_2_4_0_0

The LCOEF_2_4 coefficients are {279, -3477, 19463, 19463, -3477, 279}; Given the symmetry, we only use 3 registers.

Farrow Filter Fractional

Offset: 0a0h | Read/Write: R/W | Reset: 0b0000000100010111

Bit	Reset	Description
15:0	0x117	COEF: Coefficient 0

22.6.5.32 SPDIF_LCOEF_2_4_1_0

Offset: 0a4h | Read/Write: R/W | Reset: 0b1111001001101011

Bit	Reset	Description
15:0	0xf26b	COEF: Coefficient 1

22.6.5.33 SPDIF_LCOEF_2_4_2_0

Offset: 0a8h | Read/Write: R/W | Reset: 0b0100110000000111

Bit	Reset	Description
15:0	0x4c07	COEF: Coefficient 2

[THIS PAGE INTENTIONALLY LEFT BLANK]

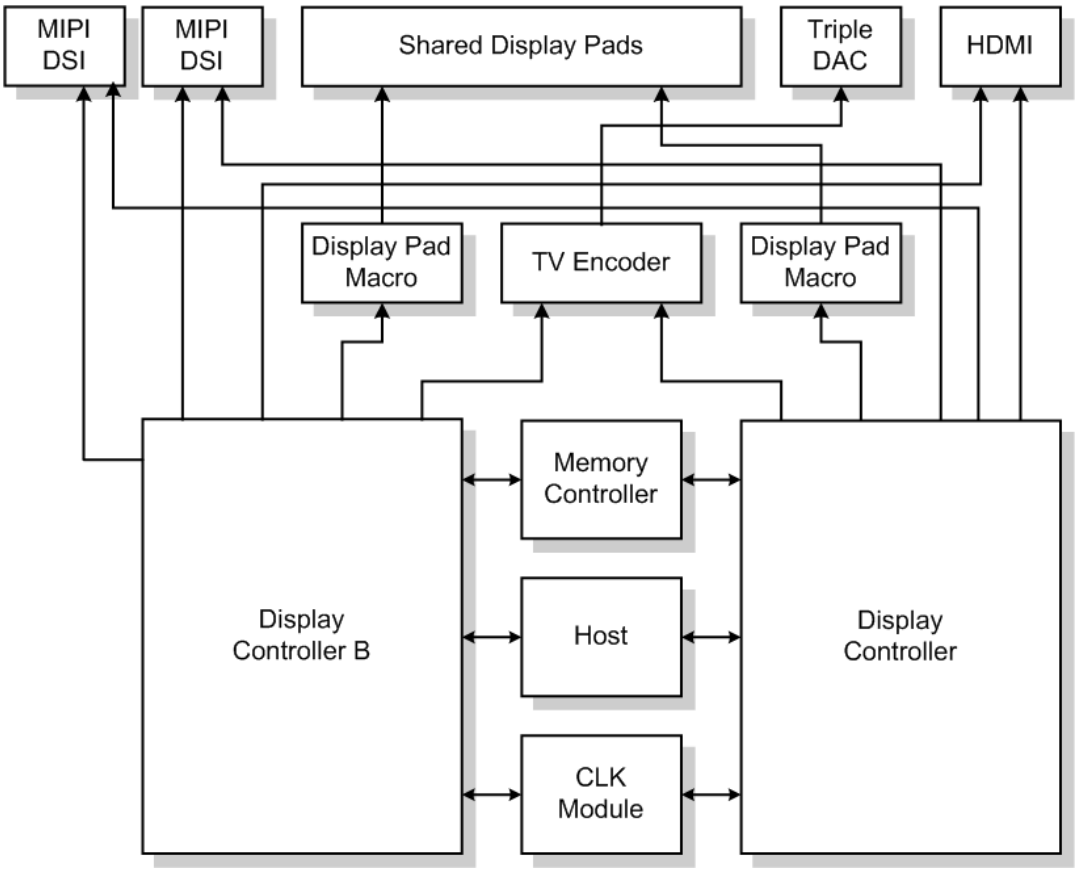
23.0 DISPLAY CONTROLLER

The Display Controller Complex integrates two independent display controllers. Each display controller is capable of interfacing to an external display device, which can be a parallel interface or SPI LCD, DVI, an HDMI® HDTV, RGB monitor or a MIPI DSI LCD. Direct interface is supported directly to most LCD displays with TFT or TFT-like interface.

There are two independent display controllers, so the module described here is instantiated twice.

Each of the two display controllers shares access to the various output resources. There are two instances of the MIPI DSI: one HDMI and one DAC. Only one display controller may access one of these outputs at any given time.

Figure 48. Display Controller



Display maximum performance is ultimately determined by the availability of memory bandwidth. The line buffer for the DSI transmitter is 1920 bytes in size. This is enough to hold 640 pixels with a pixel depth of 24-bits per pixel, or 960 pixels with a pixel depth of 16-bits per pixel. This will limit the maximum burst rate on the DSI interface for a given pixel clock frequency. However, the buffer has been sized to cope with most reasonable combinations of screen resolution, pixel clock and burst rate.

The line buffer for error diffusion dithering is limited to 1280 pixels per line. This limits the maximum horizontal active area size to 1280 pixels when error diffusion dithering is enabled. For larger displays either full 24-bit color pixel resolution output is required or ordered dithering may be used since it does not require a line buffer.

> **Note:** Display maximum resolution is not limited by counter resolution. The display raster timing generator supports 12-bit counter resolution.

Features

- Three display windows (1 graphics window A and 2 overlay windows B and C)
- Hardware surface blending
- Hardware cursor
- Fully programmable display timing and resolution
- 'Hysteresis' latency FIFO to minimize DRAM power for display refresh in an idle system

Clocking

The Display source clock can be selected from a number of sources. The SHIFT_CLK_DIVIDER register value divides the source clock to generate dcsclk, the shift clock. PIXEL_CLK_DIVIDER register value further divides down the shift to generate dcpclk, pixel clock. One pixel is processed every pixel clock and PIXEL_CLK_DIVIDER is programmed such that the correct number of shift clocks per pixel clock is available for the pixel data to be transferred to the LCD.

Display has first, second, and third level clock gating. First level clock gating is controlled by a host register bit. Display has extensive second level clock gating controlled by hardware, and based on activity.

When TVO is enabled, the TVO controls the frame rate of the Display Controller by stalling the Display Controller. TVO sends a stall signal to the Display Controller that qualifies the pixel clock enable, essentially stopping Display Controller's processing until TVO is ready for more data. When HDMI is enabled, the Display and HDMI source clocks and frequencies must be the same. Similarly, when DSI is enabled, the Display and DSI source clocks must be the same.

23.1 Hardware Interface

23.1.1 Display Pin Definition

Table 40: LCD Display Signals

Name	Description	Type
LCD_CS0_N	LCD Chip Select 0 (Main Display)	Output
LCD_CS1_N	LCD Chip Select 1(Secondary Display)	Output
LCD_D[23:0]	LCD Data	Bidirectional
LCD_DC0	LCD Serial Data/Command or Tearing Input	Bidirectional
LCD_DC1	LCD Data/Command for secondary display	Output
LCD_DE	LCD Data Enable or Tearing Input	Output
LCD_HSYNC	LCD Horizontal Sync	Bidirectional
LCD_M1	LCD Modulation	Output
LCD_PCLK	Main display RGB pixel clock or CPU write enable	Output
LCD_PWR[2:0]	LCD Power Controls	Output
LCD_SCK	Main Display LCD Serial Clock	Output
LCD_SDIN	Main Display LCD Serial Data In or Tearing Input	Input
LCD_SDOUT	Main Display LCD Serial Data Out	Output
LCD_VSYNC	LCD Vertical Sync	Output
LCD_WR_N	Secondary Display Parallel CPU Write Enable, or Serial CPU Clock	Output

Table 41: HDMI Display Signals

Name	Description	Type
HDMI_INT	Interrupt for hot-plug detect	Input
HDMI_PROBE	Reserved	Test
HDMI_RSET	Reference Set. Line to a current set resistor.	Analog
HDMI_TXCN	Transmit Clock –	Output
HDMI_TXCP	Transmit Clock +	Output
HDMI_TXD0N	Transmit (data) Lane 0 –	Output
HDMI_TXD0P	Transmit (data) Lane 0 +	Output
HDMI_TXD1N	Transmit (data) Lane 1 –	Output
HDMI_TXD1P	Transmit (data) Lane 1 +	Output
HDMI_TXD2N	Transmit (data) Lane 2 –	Output
HDMI_TXD2P	Transmit (data) Lane 2 +	Output

Table 42: TV Output CRT Display Signals

Name	Description	Type
CRT_HSYNC	Horizontal Sync	Output
CRT_VSYNC	Vertical Sync	Output
VDAC_B	Output, Blue	Analog Output
VDAC_G	Output, Green	Analog Output
VDAC_R	Output, Red	Analog Output
VDAC_RSET	Resistor Control Input	Analog
VDAC_VREF	Voltage reference input to DAC or voltage reference output	Analog

Tegra® 3 devices support a high-speed serial differential MIPI DSI interface to connect to compatible displays. Two dual lane DSI interfaces are possible. One uses the DSI A interface pins, while the other uses the CSI B interface pins, reconfigured for DSI.

Table 43: DSI-A and DSI-B Display Signals

Name	Description	Type
DSI_CLKAP	DSI Differential A Clock Positive	Output
DSI_CLKAN	DSI Differential A Clock Negative	Output
DSI_D1AP	DSI Differential A Data Lane-1 Positive	Bidirectional
DSI_D1AN	DSI Differential A Data Lane-1 Negative	Bidirectional
DSI_D2AP	DSI Differential A Data Lane-2 Positive	Output
DSI_D2AN	DSI Differential A Data Lane-2 Negative	Output
CSI_CLKBP	Optional DSI Differential B Clock Positive	Output

Name	Description	Type
CSI_CLKBN	Optional DSI Differential B Clock Negative	Output
CSI_D1BP	Optional DSI Differential B Data Lane-1 Positive	Bidirectional
CSI_D1BN	Optional DSI Differential B Data Lane-1 Negative	Bidirectional
CSI_D2BP	Optional DSI Differential B Data Lane-2 Positive	Output
CSI_D2BN	Optional DSI Differential B Data Lane-2 Negative	Output
DSI_CSI_RDN	DSI/CSI Voltage Reference Pulldown	Analog
DSI_CSI_RUP	DSI/CSI Voltage Reference Pull-up	Analog
AVDD_DSI_CSI	MIPI DSI & CSI Shared Power Rail	

23.1.2 Interface to Display

Table 44: Display Signal Interface

Pin		Primary							Secondary		
	24-bit RGB 5-wire SPI 1 clk/pixel	18-bit RGB 5-wire SPI 1 clk/pixel	16-bit RGB 5-wire SPI 1 clk/pixel	24-bit CPU 1 clk/pixel	18-bit CPU 1 clk/pixel	16-bit CPU 1 clk/pixel	9-bit CPU 2 clk/pixel 18-bpp	8-bit CPU 2 clk/pixel 16-bpp	5-wire SPI	9-bit CPU 2 clk/pixel 18-bpp	8-bit CPU 2 clk/pixel 16-bpp
LCD_D23	R1			D23						D3	D3
LCD_D22	R0			D22						D2	D2
LCD_D21	G1			D21						D7	D7
LCD_D20	G0			D20						D6	D6
LCD_D19	B1			D19							
LCD_D18	B0			D18						D0	D0
LCD_D17	R7	R5	R4	D17	D17						
LCD_D16	R6	R4	R3	D16	D16						
LCD_D15	R5	R3	R2	D15	D15	D15					
LCD_D14	R4	R2	R1	D14	D14	D14					
LCD_D13	R3	R1	R0	D13	D13	D13					
LCD_D12	R2	R0		D12	D12	D12					
LCD_D11	G7	G5	G5	D11	D11	D11					
LCD_D10	G6	G4	G4	D10	D10	D10					
LCD_D9	G5	G3	G3	D9	D9	D9					
LCD_D8	G4	G2	G2	D8	D8	D8	D8				
LCD_D7	G3	G1	G1	D7	D7	D7	D7	D7			
LCD_D6	G2	G0	G0	D6	D6	D6	D6	D6			
LCD_D5	B7	B5	B4	D5	D5	D5	D5	D5			
LCD_D4	B6	B4	B3	D4	D4	D4	D4	D4			
LCD_D3	B5	B3	B2	D3	D3	D3	D3	D3			
LCD_D2	B4	B2	B1	D2	D2	D2	D2	D2			
LCD_D1	B3	B1	B0	D1	D1	D1	D1	D1			
LCD_D0	B2	B0		D0	D0	D0	D0	D0			
LCD_PWR0									SPI_SDOUT	D4	D4
LCD_PWR1										D8	
LCD_PWR2									SPI_DIN	D5	D5
LCD_PCLK	PCLK	PCLK	PCLK	WR_	WR_	WR_	WR_	WR_			
LCD_WR_									SPI_SCK	WR_	WR_
LCD_VSYNC	VSYNC	VSYNC	VSYNC								
LCD_HSYNC	HSYNC	HSYNC	HSYNC								
LCD_DC1									DC	DC	DC
LCD_CS1_									SPI_CS_	CPU_CS_	CPU_CS_
LCD_M1										D1	D1
LCD_SCK	SPI_SCK	SPI_SCK	SPI_SCK								
LCD_SDOUT	SPI_SDOUT	SPI_SDOUT	SPI_SDOUT								
LCD_SDIN	SPI_SDIN	SPI_SDIN	SPI_SDIN								
LCD_CS0_	SPI_CS_	SPI_CS_	SPI_CS_	CPU_CS_	CPU_CS_	CPU_CS_	CPU_CS_	CPU_CS_			
LCD_DC0	DC	DC	DC	DC	DC	DC	DC	DC			
LCD_DE	DE	DE	DE	Tearing	Tearing	Tearing	Tearing	Tearing			

Note: For CPU LCDs, refer to the color data mapping below. RGB LCDs may or may not include SPI interfaces used for programming

Table 45: CPU Interface Color Mapping

Pin	24-bit, 1 clk/pixel		18-bit, 1 clk/pixel		16-bit, 1 clk/pixel		9-bit, 2 clk/pixel			8-bit, 2 clk/pixel		
	LCD Pin	LCD Color Data	LCD Pin	LCD Color Data	LCD Pin	LCD Color Data	LCD Pin	LCD Color Data Clock 1	LCD Color Data Clock 2	LCD Pin	LCD Color Data Clock 1	LCD Color Data Clock 2
LCD_D23	D23	R7										
LCD_D22	D22	R6										
LCD_D21	D21	R5										
LCD_D20	D20	R4										
LCD_D19	D19	R3										
LCD_D18	D18	R2										
LCD_D17	D17	R1	D17	R5								
LCD_D16	D16	R0	D16	R4								
LCD_D15	D15	G7	D15	R3	D15	R5						
LCD_D14	D14	G6	D14	R2	D14	R4						
LCD_D13	D13	G5	D13	R1	D13	R3						
LCD_D12	D12	G4	D12	R0	D12	R2						
LCD_D11	D11	G3	D11	G5	D11	R1						
LCD_D10	D10	G2	D10	G4	D10	G5						
LCD_D9	D9	G1	D9	G3	D9	G4						
LCD_D8	D8	G0	D8	G2	D8	G3	D8	R5	G2			
LCD_D7	D7	B7	D7	G1	D7	G2	D7	R4	G1	D7	R5	G2
LCD_D6	D6	B6	D6	G0	D6	G1	D6	R3	G0	D6	R4	G1
LCD_D5	D5	B5	D5	B5	D5	G0	D5	R2	B5	D5	R3	G0
LCD_D4	D4	B4	D4	B4	D4	B5	D4	R1	B4	D4	R2	B5
LCD_D3	D3	B3	D3	B3	D3	B4	D3	R0	B3	D3	R1	B4
LCD_D2	D2	B2	D2	B2	D2	B3	D2	G5	B2	D2	G5	B3
LCD_D1	D1	B1	D1	B1	D1	B2	D1	G4	B1	D1	G4	B2
LCD_D0	D0	B0	D0	B0	D0	B1	D0	G3	B0	D0	G3	B1

Table 46: Dual-Link RGB Interface

Pin	18-bit, 2 pixels/1 clk Dual-Link RGB	Comments
LCD_D0	ODD_B0	
LCD_D1	ODD_B1	
LCD_D2	ODD_B2	
LCD_D3	ODD_B3	
LCD_D4	ODD_B4	
LCD_D5	ODD_B5	
LCD_D6	ODD_G0	ODD Link
LCD_D7	ODD_G1	
LCD_D8	ODD_G2	
LCD_D9	ODD_G3	
LCD_D10	ODD_G4	
LCD_D11	ODD_G5	
LCD_D12	ODD_R0	

Pin	18-bit, 2 pixels/1 clk Dual-Link RGB	Comments
LCD_D12	ODD_R1	
LCD_D14	ODD_R2	
LCD_D15	ODD_R3	
LCD_D16	ODD_R4	
LCD_D17	ODD_R5	
LCD_PWR0	EVEN_B0	
LCD_PWR1	EVEN_B1	
LCD_PWR2	EVEN_B2	
LCD_WR_N	EVEN_B3	
LCD_SDIN	EVEN_B4	
LCD_D21	EVEN_B5	
LCD_D18	EVEN_G0	
LCD_D19	EVEN_G1	
LCD_DC1	EVEN_G2	
LCD_D20	EVEN_G3	EVEN Link
LCD_CS1_N	EVEN_G4	
LCD_M1	EVEN_G5	
LCD_D22	EVEN_R0	
LCD_D23	EVEN_R1	
LCD_SCK	EVEN_R2	
LCD_SDOUT	EVEN_R3	
LCD_CS0_N	EVEN_R4	
LCD_DC0	EVEN_R5	
LCD_PCLK	PCLK	165MHz Maximum
LCD_VSYNC	VSYNC	
LCD_HSYNC	HSYNC	
LCD_DE	DE	

23.1.3 Interface to Host

The display controller interfaces to the host with a similar architecture as other modules. Display's host interface contains a read FIFO and a write FIFO, allowing an asynchronous boundary between host clock and display clock. Display registers are synchronously written and read in display's clock domain. Class and register writes are treated identically.

Display controller supports several different sync points, allowing software to synchronize events properly. Refer to the display specification files for the sync points that are supported.

Display controller supports several interrupts that can be used for synchronization or for notifying the host of corruption (underflow/overflow). All interrupts are programmable to be level or edge sensitive. Refer to the Interrupt Controllers section for the interrupts that are supported.

23.1.4 Interface to Memory

Display has five memory clients: one block read client for window A (128-bit x 128 deep read FIFO), two block read clients for window B (128-bit x 64-deep read FIFO), one block read client for window C (128-bit x 128 deep read FIFO), and one single read client for the cursor. For display controller purposes, the block read client is used as a line read since display controller needs data on a line by line basis to do vertical scaling. In addition, the read memory clients for window B and C support reading YUV data.

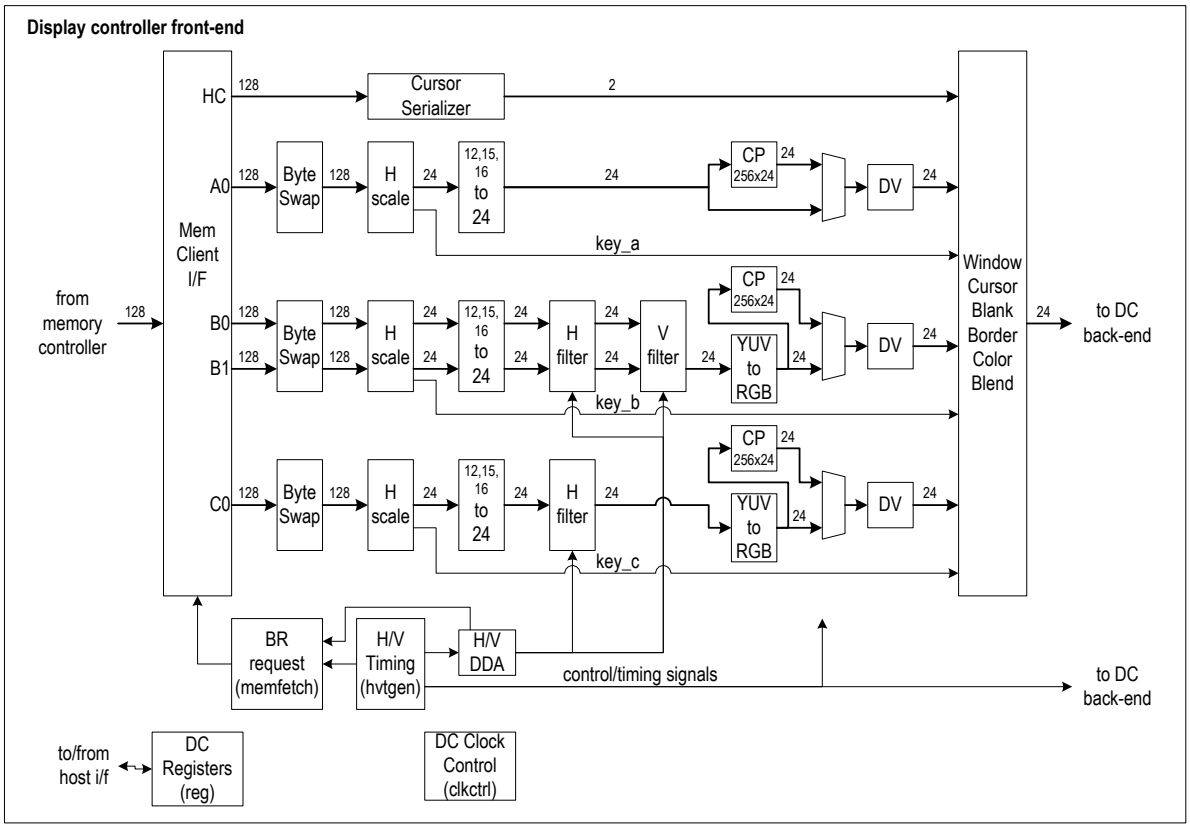
Display controller requires a constant bandwidth and can withstand only a limited latency. For this reason, display controller has the highest priority for memory controller requests. An underflow in memory data will result in a visual corruption on the LCD panel.

23.2 Functionality

23.2.1 Block Diagram

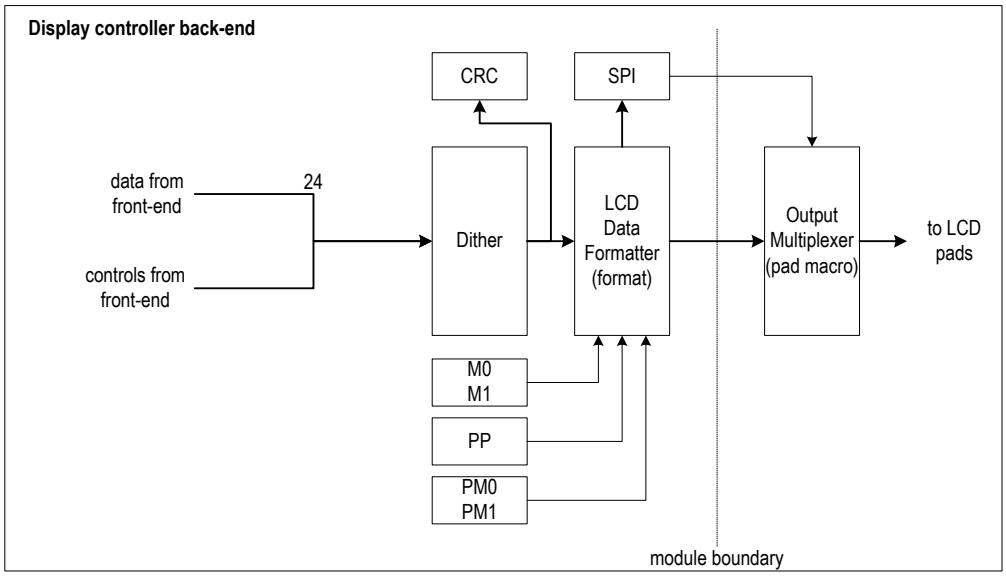
The following shows a block diagram of the display controller module front-end:

Figure 49. Display Controller Front-End Block Diagram



A block diagram of the display controller module back-end is shown in the next figure:

Figure 50. Display Controller Back-end Block Diagram



23.2.2 Color Key and Overlay Blend

When more than one active window is overlapping, the color key multiplexer selects between blended window data and pure window data. There are 3 windows (A, B, C) and therefore there are potentially 3 regions (A, B, C) where there is no window overlap and 4 regions (AB, AC, BC, ABC) where there is window overlap.

Blended data is the sum of the weighted active window data. The weight is determined from register programming and is based on a color key match, a fixed weight, an alpha weight, or a weight dependent on the other active window(s). There are separate registers defined for each possible overlap condition.

Color key can be defined in any of the overlapping window. Typically color key is enabled in one of the overlapping windows only. If color key is enabled in more than one of the overlapping windows then the color key multiplexer uses a priority encoder to select the window to use for color key compare. The order of priority is Window A, then B, then C when multiple windows are enabled for color key.

Figure 51 Keying Example with Key Region Values

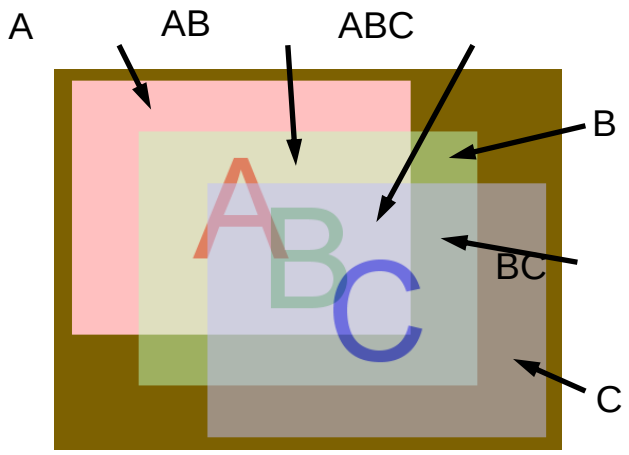
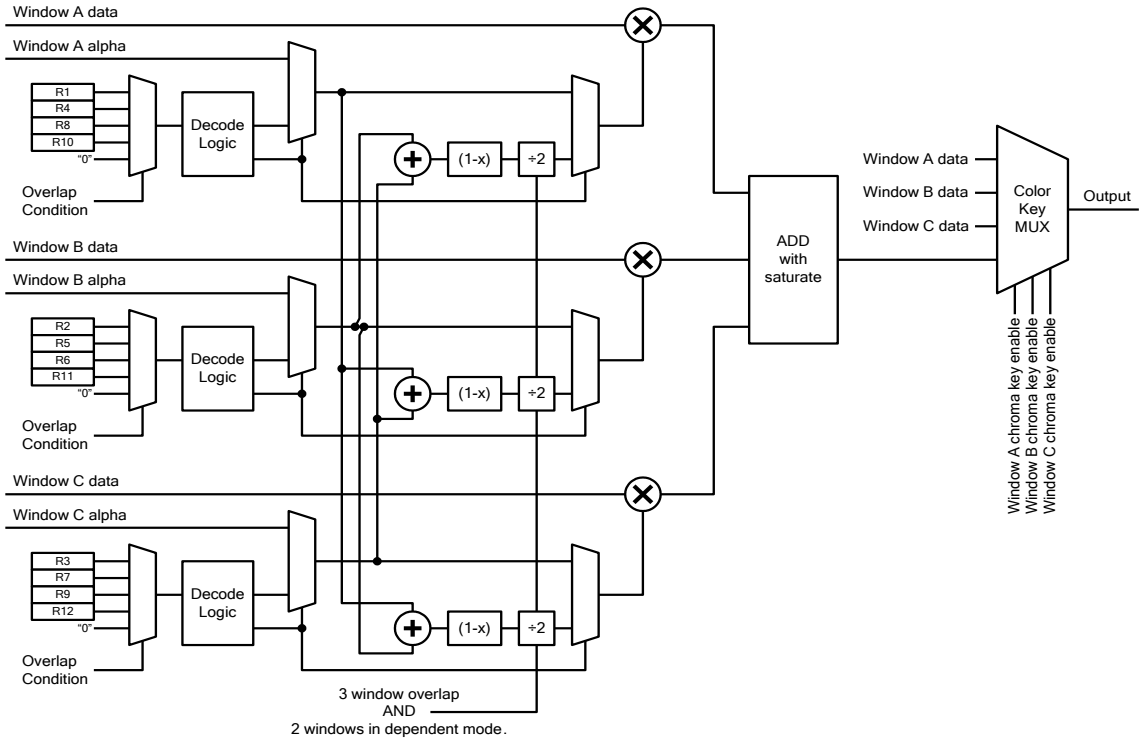


Table 47: Key Region Values for Keying Example

Region	A	B	C
A	100		
B		50	
C			50
AB	50	50	
AC	n/a	n/a	n/a
BC		25	50
ABC	25	25	50

Figure 52. Block Diagram of the Key Logic



23.2.3 Display Transformation

The Display is responsible for incrementing (and/or) decrementing *x*-direction scanning, and incrementing (and/or) decrementing *y*-direction scanning. By itself, the Display is not capable of doing the line scanning in the *y*-direction, which is needed for 90-degree or 270-degree display rotation.

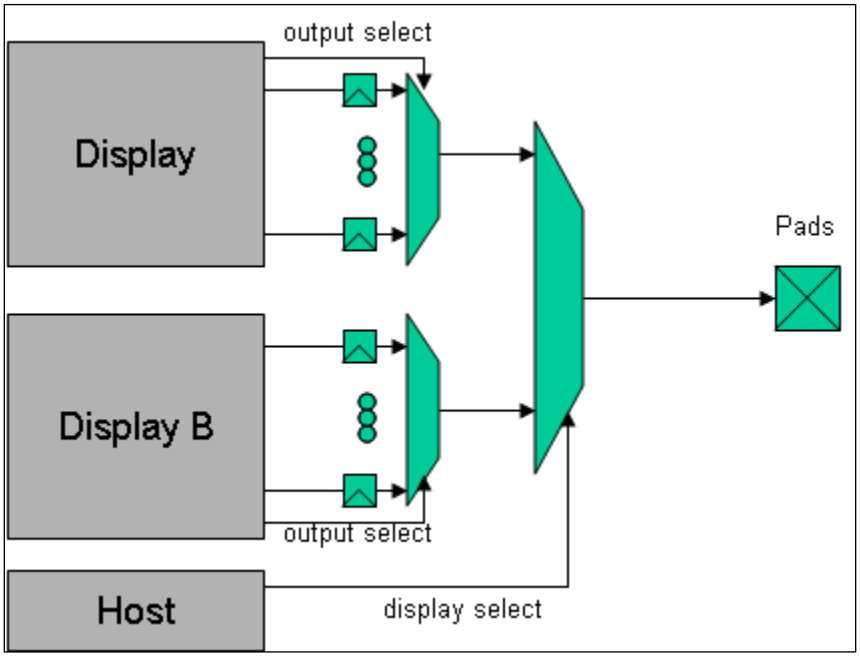
23.2.4 Dual Display

There are two identical Display controllers, Display and Display B, instantiated in the chip, allowing the chip to drive two panels simultaneously. Because of pad limitation, Display B only supports a subset of the LCD interfaces that Display supports when used simultaneously. In dual display mode, Display and Display B may be programmed to non 1-clock/1-pixel modes such that the maximum data pins needed is 9-bit (refer to table).

For dual LCD display, there are two identical pad macros, one for each instance of display. By duplicating the pad macro, each display can drive the same type of interfaces as long as the pin is available. Register bits from within the host control whether the pin is driven by Display or Display B.

Each display controller may drive any of the available outputs, allowing such combinations as (but not limited to) LCD + TV or LCD + DSI.

Figure 53. Dual Display Block Diagram



The next table shows which pairs of primary and secondary LCDs support independent simultaneous refresh of both LCDs.

Table 48: Display Pairs

Primary	Secondary	Simultaneous Dual Refresh?	Primary	Secondary	Simultaneous Dual Refresh?	Primary	Secondary	Simultaneous Dual Refresh?
24-bit RGB	9-bit CPU	No	16-bit RGB	9-bit CPU	Yes	9-bit CPU	9-bit CPU	Yes
	8-bit CPU	No		8-bit CPU	Yes		8-bit CPU	Yes
	5-wire SPI	Yes		5-wire SPI	Yes		5-wire SPI	Yes
24-bit RGB +5-wire SPI	9-bit CPU	No	16-bit RGB +5-wire SPI	9-bit CPU	Yes	8-bit CPU	9-bit CPU	Yes
	8-bit CPU	No		8-bit CPU	Yes		8-bit CPU	Yes
	5-wire SPI	Yes		5-wire SPI	Yes		5-wire SPI	Yes
18-bit RGB	9-bit CPU	Yes	18-bit CPU	9-bit CPU	Yes			
	8-bit CPU	Yes		8-bit CPU	Yes			
	5-wire SPI	Yes		5-wire SPI	Yes			
18-bit RGB +5-wire SPI	9-bit CPU	Yes	16-bit CPU	9-bit CPU	Yes			
	8-bit CPU	Yes		8-bit CPU	Yes			
	5-wire SPI	Yes		5-wire SPI	Yes			

23.3 VESA Timings

The VESA standard timings for Personal Computer Monitors can be roughly divided – for practical purposes – into two distinct groups: “Standard” and “Reduced Blanking”. Standard timing is used for CRT monitors and LCD panels. Reduced Blanking timings are used only for LCD panels that can accept Reduced Blanking timings.

The purpose of Reduced Blanking is to acknowledge the redundancy of the requirement of a large Horizontal Blanking period when outputting a video raster to an LCD panel. The Original purpose of the HB period was to allow a CRT time to move the electron beam from the right side of the screen to the left side of the screen prior to starting the next line. This takes a finite amount of time.

However, in an LCD, this time can be made arbitrarily small with suitable care in the design of the panel. The advantage that Reduced Blanking gives is that the pixel clock frequency can be reduced for a given H. Total, V. Total and Vertical Frequency (refresh rate). This in turn results in the ability to either lower power consumption for a given resolution, or increase the resolution without exceeding the capabilities of the interface between the Display Controller and the Display Device. Both sets of timing parameters are provided in the following sections.

23.4 Programming

23.4.1 Raster Generator

The following is a list of register names that correspond to the abbreviations used throughout this document.

Table 49: Abbreviation / Register Field Mapping

Abbreviation	Method/Register	Field
HFP	FRONT_PORCH	H_FRONT_PORCH
HS	SYNC_WIDTH	H_SYNC_WIDTH
HBP	BACK_PORCH	H_BACK_PORCH
HA	DISP_ACTIVE	H_DISP_ACTIVE
VFP	FRONT_PORCH	V_FRONT_PORCH
VS	SYNC_WIDTH	V_SYNC_WIDTH
VBP	BACK_PORCH	V_BACK_PORCH
VA	DISP_ACTIVE	V_DISP_ACTIVE

HT and VT are derived from the other registers by summing together all the related periods:

HT = HFP + HS + HBP + HA

VT = VFP + VS + VBP + VA

23.5 Display Controller Register Definition

Display supports three windows: Window A, Window B, Window C

Window A is a graphics only window (does not support YUV data format) and it supports only simple non-filtered up/down scaling via pixel and line replication/deletion.

Window B is an overlay window which supports both RGB and YUV data format and it supports filtered up/down scaling. Vertical filter is 2 tap with 16-phase resolution and horizontal filter is 6-tap with 16-phase resolution. Performance is optimized for YUV formats, and has performance limitations at 32-bit RGB formats.

Window C is an overlay window which supports both RGB and YUV data format and it supports filtered up/down scaling. Horizontal filter is 6-tap with 16-phase resolution.

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. Each copy is available at two different addresses. When using window address offset 0x500 (via the ardisplay_a, b, c header files), 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. When using window-direct address ranges at 0xa00, 0xc00, and 0xe00 (via the ardisplay_ad, bd, cd header files), DISPLAY_WINDOW_HEADER is not used. Each window has a separate address.

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-words dual-port register file (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.

23.5.1 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.

23.5.2 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, 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.

(TVO is an exception. Since TV does not have internal frame buffer, TVO works in continuous mode and controls the pace. DISPLAY is thus set to ONE-SHOT mode to work as slave of TVO. Other than that, it works much like CONTINUOUS mode.)

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.
- TVO Triggers

  When TVO is enabled, DISPLAY waits until TVO sends over a vsync pulse, then sends a new frame, regardless of whether an input trigger has happened.
- Tear Effect Signal Triggers

  When MSF/SSF register field is set to ENABLE, DISPLAY holds off a frame requested by input triggers until DISPALY 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. That is different from the behavior of TVO triggers.

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 behaves always 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. TVO_ENABLE and MSF/SSF_ENABLE should never be set in CONTINUOUS mode.

23.5.3 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.

23.5.4 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

23.6 Display CMD Registers

23.6.1 DC_CMD_GENERAL_INCR_SYNCPT_0

Offset: 000h | Read/Write: R/W | Reset: 0b0000000000000000

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

23.6.2 DC_CMD_GENERAL_INCR_SYNCPT_CNTRL_0

Offset: 001h | Read/Write: R/W | Reset: 0b0xxxxxxx0

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 soft reset, first set SOFT_RESET of all host1x clients affected, then clear all SOFT_RESETs.

23.6.3 DC_CMD_GENERAL_INCR_SYNCPT_ERROR_0

Offset: 002h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	_NONE_	GENERAL_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows This bit is sticky and will remain set until overwritten with a zero

23.6.4 DC_CMD_WIN_A_INCR_SYNCPT_0

Offset: 008h | Read/Write: R/W | Reset: 0b0000000000000000

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

23.6.5 DC_CMD_WIN_A_INCR_SYNCPT_CNTRL_0

Offset: 009h | Read/Write: R/W | Reset: 0b0xxxxxx0

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 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.

23.6.6 DC_CMD_WIN_A_INCR_SYNCPT_ERROR_0

Offset: 00ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	_NONE_	WIN_A_COND_STATUS: COND_STATUS[COND] is set if the fifo for COND overflows This bit is sticky and will remain set until overwritten with a zero

23.6.7 DC_CMD_WIN_B_INCR_SYNCPT_0

Offset: 010h | Read/Write: R/W | Reset: 0b0000000000000000

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

23.6.8 DC_CMD_WIN_B_INCR_SYNCPT_CNTRL_0

Offset: 011h | Read/Write: R/W | Reset: 0b0xxxxxx0

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 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.

23.6.9 DC_CMD_WIN_B_INCR_SYNCPT_ERROR_0

Offset: 012h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	_NONE_	WIN_B_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows This bit is sticky and will remain set until overwritten with a zero

23.6.10 DC_CMD_WIN_C_INCR_SYNCPT_0

Offset: 018h | Read/Write: R/W | Reset: 0b0000000000000000

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

23.6.11 DC_CMD_WIN_C_INCR_SYNCPT_CNTRL_0

Offset: 019h | Read/Write: R/W | Reset: 0b0xxxxxxx0

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 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.

23.6.12 DC_CMD_WIN_C_INCR_SYNCPT_ERROR_0

Offset: 01ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	_NONE_	WIN_C_COND_STATUS: COND_STATUS[COND] is set if the FIFO for COND overflows This bit is sticky and will remain set until overwritten with a zero

23.6.13 DC_CMD_CONT_SYNCPT_VSYNC_0

Offset: 028h | Read/Write: R/W | Reset: 0b0xxxxxxxx

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	_NONE_	VSYNC_INDX: return INDX (set HOST_CLRD packet TYPE field to SYNCPT)

23.6.14 DC_CMD_CTXSW_0

Context switch registers for class and channel

Should be common to all modules. Includes the current channel/class (which is writable by SW) and the next channel/class (which the hardware sets when it receives a context switch).

Context switching 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. SW 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: 030h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxx11111x0000000000

Bit	Reset	Description
31:28	0xf	NEXT_CHANNEL: Next requested channel
25:16	0x0	NEXT_CLASS: Next requested class
15:12	0xf	CURR_CHANNEL: Current working channel, reset to 'invalid'
11	0x1	AUTO_ACK: Automatically acknowledge any incoming context switch requests 0 = MANUAL 1 = AUTOACK
9:0	0x0	CURR_CLASS: Current working class

23.6.15 DC_CMD_DISPLAY_COMMAND_OPTION0_0

Display Controller Option 0. This register is not effective until DISPLAY_COMMAND is written.

Class: Display Command

Offset: 031h | Read/Write: R/W | Reset: 0b000xxxxxxxx00000000

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 reflects 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 0= Disabled 1= Enabled 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 0= Disabled 1= Enabled When enabled, MSF signal can be input through 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

23.6.16 DC_CMD_DISPLAY_COMMAND_0

Display Command

Offset: 032h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxx00xxxx0

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

23.6.17 DC_CMD_SIGNAL_RAISE_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_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: 033h | Read/Write: R/W | Reset: 0bxxxxxxx0xxxxxxxxxxx

Bit	Reset	Description
19:16	X	SIGNAL_RAISE_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE_TYPE: 0= Oneshot, 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

23.6.18 DC_CMD_DISPLAY_POWER_CONTROL_0

Display Power Control

Offset: 036h | Read/Write: R/W | Reset: 0b00xxxxx0x0xxxxxxx0x0x0x0x0

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
2	0x0	PW1_ENABLE: PW1 signal Enable. This signal can be output at the pad for display power sequencing. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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

23.6.19 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: 037h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
20	X	GPIO_2_INT: GPIO 2 Interrupt Status, connected to LCD_PWR2 0= interrupt not pending 1= interrupt pending
19	X	GPIO_1_INT: GPIO 1 Interrupt Status, connected to LCD_PWR1 0= interrupt not pending 1= interrupt pending
18	X	GPIO_0_INT: GPIO 0 Interrupt Status, connected to LCD_PWR0 0= interrupt not pending 1= interrupt pending
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
11	X	EPP_OF_INT: Display2epp Overflow 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

Bit	Reset	Description
9	X	WIN_B_UF_INT: Window B Underflow Interrupt Status 0= interrupt not pending 1= interrupt pending
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
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

23.6.20 DC_CMD_INT_MASK_0

Setting bits in this register masked 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.

Interrupt Mask

Offset: 038h | Read/Write: R/W | Reset: 0b000x00000x0000xx00000

Bit	Reset	Description
20	0x0	GPIO_2_INT_MASK: GPIO 2 Interrupt Mask, connected to LCD_PWR2 0 = MASKED 1 = NOTMASKED
19	0x0	GPIO_1_INT_MASK: GPIO 1 Interrupt Mask, connected to LCD_PWR1 0 = MASKED 1 = NOTMASKED
18	0x0	GPIO_0_INT_MASK: GPIO 0 Interrupt Mask, connected to LCD_PWR0 0= interrupt masked 1= interrupt not masked 0 = MASKED 1 = NOTMASKED

Bit	Reset	Description
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
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
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

23.6.21 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: 039h | Read/Write: R/W | Reset: 0b000x00000x0000xx00001

Bit	Reset	Description
20	0x0	GPIO_2_INT_ENABLE: Display GPIO_2 Interrupt Enable, connected to LCD_PWR2 0 = DISABLE 1 = ENABLE
19	0x0	GPIO_1_INT_ENABLE: Display GPIO_1 Interrupt Enable, connected to LCD_PWR1 0 = DISABLE 1 = ENABLE
18	0x0	GPIO_0_INT_ENABLE: Display GPIO_0 Interrupt Enable, connected to LCD_PWR0 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

Bit	Reset	Description
4	0x0	V_PULSE3_INT_ENABLE: Vertical Pulse 3 Interrupt Enable 0= interrupt masked 1= interrupt not masked 0 = DISABLE 1 = ENABLE
3	0x0	H_BLANK_INT_ENABLE: Horizontal Blank Interrupt Enable 0= interrupt masked 1= interrupt not masked 0 = DISABLE 1 = ENABLE
2	0x0	V_BLANK_INT_ENABLE: Vertical Blank Interrupt Enable 0= interrupt masked 1= interrupt not masked 0 = DISABLE 1 = ENABLE
1	0x0	FRAME_END_INT_ENABLE: Frame End Interrupt Enable 0= interrupt masked 1= interrupt not masked 0 = DISABLE 1 = ENABLE
0	0x1	CTXSW_INT_ENABLE: Context Switch Interrupt Enable 0= interrupt disabled 1= interrupt enabled 0 = DISABLE 1 = ENABLE

23.6.22 DC_CMD_INT_TYPE_0

Two interrupt types are available: a. Edge interrupt - transition on input signal/event generates pending interrupt b. Level interrupt - active level on input signal/event generates pending interrupt

Interrupt Type

Offset: 03ah | Read/Write: R/W | Reset: 0b000x00000x0000xx0000

Bit	Reset	Description
20	0x0	GPIO_2_INT_TYPE: Display GPIO_2 Interrupt Type, connected to LCD_PWR2 0 = EDGE 1 = LEVEL
19	0x0	GPIO_1_INT_TYPE: Display GPIO_1 Interrupt Type, connected to LCD_PWR1 0 = EDGE 1 = LEVEL
18	0x0	GPIO_0_INT_TYPE: Display GPIO_0 Interrupt Type, connected to LCD_PWR0 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

Bit	Reset	Description
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
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
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

23.6.23 DC_CMD_INT_POLARITY_0

For edge interrupt, these bits specify whether a pending interrupt is generated on falling edge or on rising edge of the corresponding input signal/event. For level interrupt, these bits specify whether a pending interrupt is generated on low level or on 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: 03bh | Read/Write: R/W | Reset: 0b000x00000x0000xx0000

Bit	Reset	Description
20	0x0	GPIO_2_INT_POLARITY: Display GPIO_2 Interrupt. Interrupt Polarity, connected to LCD_PWR2 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH

Bit	Reset	Description
19	0x0	GPIO_1_INT_POLARITY: Display GPIO_1 Interrupt. Interrupt Polarity, connected to LCD_PWR1 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
18	0x0	GPIO_0_INT_POLARITY: Display GPIO_0 Interrupt. Interrupt Polarity, connected to LCD_PWR0 0= falling edge or low level interrupt 1= rising edge or high level interrupt 0 = LOW 1 = HIGH
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
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

Bit	Reset	Description
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

23.6.24 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: 03ch | Read/Write: R/W | Reset: 0bxxxxxxx0xxxxxxxxxxx

Bit	Reset	Description
19:16	X	SIGNAL_RAISE1_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE1_TYPE: 0= Oneshot, 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
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

23.6.25 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: 03dh | Read/Write: R/W | Reset: 0bxxxxxxx0xxxxxxxxxxx

Bit	Reset	Description
19:16	X	SIGNAL_RAISE2_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE2_TYPE: 0= Oneshot, 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

Bit	Reset	Description
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

23.6.26 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: 03eh | Read/Write: R/W | Reset: 0bxxxxxxx0xxxxxxxxxxx

Bit	Reset	Description
19:16	X	SIGNAL_RAISE3_CHANNEL_ID: Signal Raise Channel ID
12	0x0	SIGNAL_RAISE3_TYPE: 0= Oneshot, 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

23.6.27 DC_CMD_STATE_ACCESS_0

Double/triple buffers read and write access control

Offset: 040h | Read/Write: R/W | Reset: 0b0x0

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

23.6.28 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: 041h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxx0000xxxx0000

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 req pending/req completed 1= activation requested/pending 0 = DISABLE 1 = ENABLE
2	0x0	WIN_B_ACT_REQ: Window B activation request 0= no req pending/req completed 1= activation requested/pending 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	0x0	WIN_A_ACT_REQ: Window A activation request 0= no req pending/req completed 1= activation requested/pending 0 = DISABLE 1 = ENABLE
0	0x0	GENERAL_ACT_REQ: Non-window-specific 0= no req pending/req completed 1= activation requested/pending 0 = DISABLE 1 = ENABLE

23.6.29 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: 042h | Read/Write: R/W | Reset: 0b000

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

23.6.30 DC_CMD_REG_ACT_CONTROL_0

Register activation options

Offset: 043h | Read/Write: R/W | Reset: 0b0x0x0x0

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
2	0x0	WIN_A_ACT_CNTR_SEL: Select which counter to use for window A activation 0 = VCOUNTER 1 = HCOUNTER

Bit	Reset	Description
0	0x0	GENERAL_ACT_CNTR_SEL: Select which counter to use for general activation 0 = VCOUNTER 1 = HCOUNTER

23.7 Display COM Registers

23.7.1 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_INTPU_DATA = FULL_FRAME, DISPLAY_COMMAND.DISPLAY_CTRL_MODE should be programmed to C_DISPLAY so that CRC works properly.

When CRC_INTPU_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-bit 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 {LCD_D20, LPV0, LCD_D19, LCD_D18, LCD_D21, 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: 300h | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3	0x0	CRC_ALWAYS: CRC always: calculate 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

23.7.2 DC_COM_CRC_CHECKSUM_0

CRC Checksum

This register can be read by host after CRC logic stops capturing data.

Offset: 301h | Read/Write: RO | Reset: 0x00000000

Bit	Reset	Description
31:0	0x0	CRC_CHECKSUM: CRC Checksum

23.7.3 DC_COM_PIN_OUTPUT_ENABLE0_0

Pin Output Enable 0

Pin Output Enable registers

Offset: 302h | Read/Write: R/W | Reset: 0b0x0x0x0x0x0x0x0x0x0x0x0x0x0x0x0x

Bit	Reset	Description
30	0x0	LD15_OUTPUT_ENABLE: LD15 pin output enable 0 = ENABLE 1 = DISABLE
28	0x0	LD14_OUTPUT_ENABLE: LD14 pin output enable 0 = ENABLE 1 = DISABLE
26	0x0	LD13_OUTPUT_ENABLE: LD13 pin output enable 0 = ENABLE 1 = DISABLE
24	0x0	LD12_OUTPUT_ENABLE: LD12 pin output enable 0 = ENABLE 1 = DISABLE
22	0x0	LD11_OUTPUT_ENABLE: LD11 pin output enable 0 = ENABLE 1 = DISABLE
20	0x0	LD10_OUTPUT_ENABLE: LD10 pin output enable 0 = ENABLE 1 = DISABLE
18	0x0	LD9_OUTPUT_ENABLE: LD9 pin output enable 0 = ENABLE 1 = DISABLE
16	0x0	LD8_OUTPUT_ENABLE: LD8 pin output enable 0 = ENABLE 1 = DISABLE
14	0x0	LD7_OUTPUT_ENABLE: LD7 pin output enable 0 = ENABLE 1 = DISABLE
12	0x0	LD6_OUTPUT_ENABLE: LD6 pin output enable 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
10	0x0	LD5_OUTPUT_ENABLE: LD5 pin output enable 0 = ENABLE 1 = DISABLE
8	0x0	LD4_OUTPUT_ENABLE: LD4 pin output enable 0 = ENABLE 1 = DISABLE
6	0x0	LD3_OUTPUT_ENABLE: LD3 pin output enable 0 = ENABLE 1 = DISABLE
4	0x0	LD2_OUTPUT_ENABLE: LD2 pin output enable 0 = ENABLE 1 = DISABLE
2	0x0	LD1_OUTPUT_ENABLE: LD1 pin output enable 0 = ENABLE 1 = DISABLE
0	0x0	LD0_OUTPUT_ENABLE: LD0 pin output enable 0 = ENABLE 1 = DISABLE

23.7.4 DC_COM_PIN_OUTPUT_ENABLE1_0

Pin Output Enable 1

Offset: 303h | Read/Write: R/W | Reset: 0b0x0x0x0xxx0x0x0xxxxxxxxxxxxx0x0

Bit	Reset	Description
30	0x0	LHS_OUTPUT_ENABLE: LCD_HSYNC pin output enable 0 = ENABLE 1 = DISABLE
28	0x0	LVS_OUTPUT_ENABLE: LVS pin output enable 0 = ENABLE 1 = DISABLE
26	0x0	LSC1_OUTPUT_ENABLE: LCD_WR_N pin output enable 0 = ENABLE 1 = DISABLE
24	0x0	LSC0_OUTPUT_ENABLE: LCD_PCLK pin output enable 0 = ENABLE 1 = DISABLE
20	0x0	LPW2_OUTPUT_ENABLE: LCD_PWR2 pin output enable 0 = ENABLE 1 = DISABLE
18	0x0	LPW1_OUTPUT_ENABLE: LCD_PWR1 pin output enable 0 = ENABLE 1 = DISABLE
16	0x0	LPW0_OUTPUT_ENABLE: LCD_PWR0 pin output enable 0 = ENABLE 1 = DISABLE

Bit	Reset	Description
2	0x0	LD17_OUTPUT_ENABLE: LD17 pin output enable 0 = ENABLE 1 = DISABLE
0	0x0	LD16_OUTPUT_ENABLE: LD16 pin output enable 0 = ENABLE 1 = DISABLE

23.7.5 DC_COM_PIN_OUTPUT_ENABLE2_0

Pin Output Enable 2

Offset: 304h | Read/Write: R/W | Reset: 0b1x1x0x1xxxxx0x1xxx0x1x0

Bit	Reset	Description
22	0x1	LPP_OUTPUT_ENABLE: LCD_D23 pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
20	0x1	LDI_OUTPUT_ENABLE: LCD_D22 pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
18	0x0	LM1_OUTPUT_ENABLE: LCD_M1 pin output enable 0 = ENABLE 1 = DISABLE
16	0x1	LM0_OUTPUT_ENABLE: LCD_CS1_N pin output enable 0 = ENABLE 1 = DISABLE
10	0x0	LVP1_OUTPUT_ENABLE: LCD_D20 pin output enable 0 = ENABLE 1 = DISABLE
8	0x1	LVP0_OUTPUT_ENABLE: LCD_DC1 pin output enable 0 = ENABLE 1 = DISABLE
4	0x0	LHP2_OUTPUT_ENABLE: LCD_D19 pin output enable 0 = ENABLE 1 = DISABLE
2	0x1	LHP1_OUTPUT_ENABLE: LCD_D18 pin output enable 0 = ENABLE 1 = DISABLE
0	0x0	LHP0_OUTPUT_ENABLE: LCD_D21 pin output enable 0 = ENABLE 1 = DISABLE

23.7.6 DC_COM_PIN_OUTPUT_ENABLE3_0

Pin Output Enable 3

Offset: 305h | Read/Write: R/W | Reset: 0b1x1x1x1x1x1

Bit	Reset	Description
10	0x1	LSDI_OUTPUT_ENABLE: LSDI pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
8	0x1	LSPI_OUTPUT_ENABLE: LSPI pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
6	0x1	LDC_OUTPUT_ENABLE: LDC pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
4	0x1	LCSN_OUTPUT_ENABLE: LCSN pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
2	0x1	LSDA_OUTPUT_ENABLE: LSDA pin output enable 0 = ENABLE 1 = DISABLE (default after reset)
0	0x1	LSCK_OUTPUT_ENABLE: LSCK pin output enable 0 = ENABLE 1 = DISABLE (default after reset)

23.7.7 DC_COM_PIN_OUTPUT_POLARITY0_0

Pin Output Polarity 0

Pin Output Polarity registers

Offset: 306h | Read/Write: R/W | Reset: 0b0x0x0x0x0x0x0x0x0x0x0x0x0x0x0x0x

Bit	Reset	Description
30	0x0	LD15_OUTPUT_POLARITY: LD15 pin output polarity 0 = HIGH 1 = LOW
28	0x0	LD14_OUTPUT_POLARITY: LD14 pin output polarity 0 = HIGH 1 = LOW
26	0x0	LD13_OUTPUT_POLARITY: LD13 pin output polarity 0 = HIGH 1 = LOW
24	0x0	LD12_OUTPUT_POLARITY: LD12 pin output polarity 0 = HIGH 1 = LOW
22	0x0	LD11_OUTPUT_POLARITY: LD11 pin output polarity 0 = HIGH 1 = LOW

Bit	Reset	Description
20	0x0	LD10_OUTPUT_POLARITY: LD10 pin output polarity 0 = HIGH 1 = LOW
18	0x0	LD9_OUTPUT_POLARITY: LD9 pin output polarity 0 = HIGH 1 = LOW
16	0x0	LD8_OUTPUT_POLARITY: LD8 pin output polarity 0 = HIGH 1 = LOW
14	0x0	LD7_OUTPUT_POLARITY: LD7 pin output polarity 0 = HIGH 1 = LOW
12	0x0	LD6_OUTPUT_POLARITY: LD6 pin output polarity 0 = HIGH 1 = LOW
10	0x0	LD5_OUTPUT_POLARITY: LD5 pin output polarity 0 = HIGH 1 = LOW
8	0x0	LD4_OUTPUT_POLARITY: LD4 pin output polarity 0 = HIGH 1 = LOW
6	0x0	LD3_OUTPUT_POLARITY: LD3 pin output polarity 0 = HIGH 1 = LOW
4	0x0	LD2_OUTPUT_POLARITY: LD2 pin output polarity 0 = HIGH 1 = LOW
2	0x0	LD1_OUTPUT_POLARITY: LD1 pin output polarity 0 = HIGH 1 = LOW
0	0x0	LD0_OUTPUT_POLARITY: LD0 pin output polarity 0 = HIGH 1 = LOW

23.7.8 DC_COM_PIN_OUTPUT_POLARITY1_0

Pin Output Polarity 1

Offset: 307h | Read/Write: R/W | Reset: 0b0x0x0x0xxx0x0x0xxxxxxxxxxxxx0x0

Bit	Reset	Description
30	0x0	LHS_OUTPUT_POLARITY: LCD_HSYNC pin output polarity 0 = HIGH 1 = LOW
28	0x0	LVS_OUTPUT_POLARITY: LVS pin output polarity 0 = HIGH 1 = LOW
26	0x0	LSC1_OUTPUT_POLARITY: LCD_WR_N pin output polarity 0 = HIGH 1 = LOW

Bit	Reset	Description
24	0x0	LSC0_OUTPUT_POLARITY: LCD_PCLK pin output polarity 0 = HIGH 1 = LOW
20	0x0	LPW2_OUTPUT_POLARITY: LCD_PWR2 pin output polarity 0 = HIGH 1 = LOW
18	0x0	LPW1_OUTPUT_POLARITY: LCD_PWR1 pin output polarity 0 = HIGH 1 = LOW
16	0x0	LPW0_OUTPUT_POLARITY: LCD_PWR0 pin output polarity 0 = HIGH 1 = LOW
2	0x0	LD17_OUTPUT_POLARITY: LD17 pin output polarity 0 = HIGH 1 = LOW
0	0x0	LD16_OUTPUT_POLARITY: LD16 pin output polarity 0 = HIGH 1 = LOW

23.7.9 DC_COM_PIN_OUTPUT_POLARITY2_0

Pin Output Polarity 2

Offset: 308h | Read/Write: R/W | Reset: 0b0x0x0x0xxxxx0x0xxx0x0x0

Bit	Reset	Description
22	0x0	LPP_OUTPUT_POLARITY: LCD_D23 pin output polarity 0 = HIGH 1 = LOW
20	0x0	LDI_OUTPUT_POLARITY: LCD_D22 pin output polarity 0 = HIGH 1 = LOW
18	0x0	LM1_OUTPUT_POLARITY: LCD_M1 pin output polarity 0 = HIGH 1 = LOW
16	0x0	LM0_OUTPUT_POLARITY: LCD_CS1_N pin output polarity 0 = HIGH 1 = LOW
10	0x0	LVP1_OUTPUT_POLARITY: LCD_D20 pin output polarity 0 = HIGH 1 = LOW
8	0x0	LVP0_OUTPUT_POLARITY: LCD_DC1 pin output polarity 0 = HIGH 1 = LOW
4	0x0	LHP2_OUTPUT_POLARITY: LCD_D19 pin output polarity 0 = HIGH 1 = LOW
2	0x0	LHP1_OUTPUT_POLARITY: LCD_D18 pin output polarity 0 = HIGH 1 = LOW

Bit	Reset	Description
0	0x0	LHP0_OUTPUT_POLARITY: LCD_D21 pin output polarity 0 = HIGH 1 = LOW

23.7.10 DC_COM_PIN_OUTPUT_POLARITY3_0

Pin Output Polarity 3

Offset: 309h | Read/Write: R/W | Reset: 0b0x0x0x0x0x0

Bit	Reset	Description
10	0x0	LSDI_OUTPUT_POLARITY: LSDI pin output polarity 0 = HIGH 1 = LOW
8	0x0	LSPI_OUTPUT_POLARITY: LSPI pin output polarity 0 = HIGH 1 = LOW
6	0x0	LDC_OUTPUT_POLARITY: LDC pin output polarity 0 = HIGH 1 = LOW
4	0x0	LCSN_OUTPUT_POLARITY: LCSN pin output polarity 0 = HIGH 1 = LOW
2	0x0	LSDA_OUTPUT_POLARITY: LSDA pin output polarity 0 = HIGH 1 = LOW
0	0x0	LSCK_OUTPUT_POLARITY: LSCK pin output polarity 0 = HIGH 1 = LOW

23.7.11 DC_COM_PIN_OUTPUT_DATA0_0

Pin Output Data 0

Pin Output data registers

To change output data, the corresponding mask should be disabled (not masked).

Offset: 30ah | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	LD15_OUTPUT_DATA_MASK: LD15 pin output data mask 0 = MASKED 1 = NOTMASKED
30	0x0	LD15_OUTPUT_DATA: LD15 pin output data 0 = LOW 1 = HIGH
29	0x0	LD14_OUTPUT_DATA_MASK: LD14 pin output data mask 0 = MASKED 1 = NOTMASKED

Bit	Reset	Description
28	0x0	LD14_OUTPUT_DATA: LD14 pin output data 0 = LOW 1 = HIGH
27	0x0	LD13_OUTPUT_DATA_MASK: LD13 pin output data mask 0 = MASKED 1 = NOTMASKED
26	0x0	LD13_OUTPUT_DATA: LD13 pin output data 0 = LOW 1 = HIGH
25	0x0	LD12_OUTPUT_DATA_MASK: LD12 pin output data 0 = MASKED 1 = NOTMASKED
24	0x0	LD12_OUTPUT_DATA: LD12 pin output data 0 = LOW 1 = HIGH
23	0x0	LD11_OUTPUT_DATA_MASK: LD11 pin output data mask 0 = MASKED 1 = NOTMASKED
22	0x0	LD11_OUTPUT_DATA: LD11 pin output data 0 = LOW 1 = HIGH
21	0x0	LD10_OUTPUT_DATA_MASK: LD10 pin output data mask 0 = MASKED 1 = NOTMASKED
20	0x0	LD10_OUTPUT_DATA: LD10 pin output data 0 = LOW 1 = HIGH
19	0x0	LD9_OUTPUT_DATA_MASK: LD9 pin output data mask 0 = MASKED 1 = NOTMASKED
18	0x0	LD9_OUTPUT_DATA: LD9 pin output data 0 = LOW 1 = HIGH
17	0x0	LD8_OUTPUT_DATA_MASK: LD8 pin output data mask 0 = MASKED 1 = NOTMASKED
16	0x0	LD8_OUTPUT_DATA: LD8 pin output data 0 = LOW 1 = HIGH
15	0x0	LD7_OUTPUT_DATA_MASK: LD7 pin output data mask 0 = MASKED 1 = NOTMASKED
14	0x0	LD7_OUTPUT_DATA: LD7 pin output data 0 = LOW 1 = HIGH
13	0x0	LD6_OUTPUT_DATA_MASK: LD6 pin output data mask 0 = MASKED 1 = NOTMASKED

Bit	Reset	Description
12	0x0	LD6_OUTPUT_DATA: LD6 pin output data 0 = LOW 1 = HIGH
11	0x0	LD5_OUTPUT_DATA_MASK: LD5 pin output data mask 0 = MASKED 1 = NOTMASKED
10	0x0	LD5_OUTPUT_DATA: LD5 pin output data 0 = LOW 1 = HIGH
9	0x0	LD4_OUTPUT_DATA_MASK: LD4 pin output data mask 0 = MASKED 1 = NOTMASKED
8	0x0	LD4_OUTPUT_DATA: LD4 pin output data 0 = LOW 1 = HIGH
7	0x0	LD3_OUTPUT_DATA_MASK: LD3 pin output data mask 0 = MASKED 1 = NOTMASKED
6	0x0	LD3_OUTPUT_DATA: LD3 pin output data 0 = LOW 1 = HIGH
5	0x0	LD2_OUTPUT_DATA_MASK: LD2 pin output data mask 0 = MASKED 1 = NOTMASKED
4	0x0	LD2_OUTPUT_DATA: LD2 pin output data 0 = LOW 1 = HIGH
3	0x0	LD1_OUTPUT_DATA_MASK: LD1 pin output data mask 0 = MASKED 1 = NOTMASKED
2	0x0	LD1_OUTPUT_DATA: LD1 pin output data 0 = LOW 1 = HIGH
1	0x0	LD0_OUTPUT_DATA_MASK: LD0 pin output data mask 0 = MASKED 1 = NOTMASKED
0	0x0	LD0_OUTPUT_DATA: LD0 pin output data 0 = LOW 1 = HIGH

23.7.12 DC_COM_PIN_OUTPUT_DATA1_0

Pin Output Data 1

Offset: 30bh | Read/Write: R/W | Reset: 0b00000000xx000000xxxxxxxxxxxx0000

Bit	Reset	Description
31	0x0	LHS_OUTPUT_DATA_MASK: LCD_HSYNC pin output data mask 0 = MASKED 1 = NOTMASKED

Bit	Reset	Description
30	0x0	LHS_OUTPUT_DATA: LCD_HSYNC pin output data 0 = LOW 1 = HIGH
29	0x0	LVS_OUTPUT_DATA_MASK: LVS pin output data mask 0 = MASKED 1 = NOTMASKED
28	0x0	LVS_OUTPUT_DATA: LVS pin output data 0 = LOW 1 = HIGH
27	0x0	LSC1_OUTPUT_DATA_MASK: LCD_WR_N pin output data mask 0 = MASKED 1 = NOTMASKED
26	0x0	LSC1_OUTPUT_DATA: LCD_WR_N pin output data 0 = LOW 1 = HIGH
25	0x0	LSC0_OUTPUT_DATA_MASK: LCD_PCLK pin output data mask 0 = MASKED 1 = NOTMASKED
24	0x0	LSC0_OUTPUT_DATA: LCD_PCLK pin output data 0 = LOW 1 = HIGH
21	0x0	LPW2_OUTPUT_DATA_MASK: LCD_PWR2 pin output data mask 0 = MASKED 1 = NOTMASKED
20	0x0	LPW2_OUTPUT_DATA: LCD_PWR2 pin output data 0 = LOW 1 = HIGH
19	0x0	LPW1_OUTPUT_DATA_MASK: LCD_PWR1 pin output data mask 0 = MASKED 1 = NOTMASKED
18	0x0	LPW1_OUTPUT_DATA: LCD_PWR1 pin output data 0 = LOW 1 = HIGH
17	0x0	LPW0_OUTPUT_DATA_MASK: LCD_PWR0 pin output data mask 0 = MASKED 1 = NOTMASKED
16	0x0	LPW0_OUTPUT_DATA: LCD_PWR0 pin output data 0 = LOW 1 = HIGH
3	0x0	LD17_OUTPUT_DATA_MASK: LD17 pin output data mask 0 = MASKED 1 = NOTMASKED
2	0x0	LD17_OUTPUT_DATA: LD17 pin output data 0 = LOW 1 = HIGH
1	0x0	LD16_OUTPUT_DATA_MASK: LD16 pin output data mask 0 = MASKED 1 = NOTMASKED

Bit	Reset	Description
0	0x0	LD16_OUTPUT_DATA: LD16 pin output data 0 = LOW 1 = HIGH

23.7.13 DC_COM_PIN_OUTPUT_DATA2_0

Pin Output Data 2

Offset: 30ch | Read/Write: R/W | Reset: 0b00000000xxxx0000xx000000

Bit	Reset	Description
23	0x0	LPP_OUTPUT_DATA_MASK: LCD_D23 pin output data mask 0 = MASKED 1 = NOTMASKED
22	0x0	LPP_OUTPUT_DATA: LCD_D23 pin output data 0 = LOW 1 = HIGH
21	0x0	LDI_OUTPUT_DATA_MASK: LCD_D22 pin output data mask 0 = MASKED 1 = NOTMASKED
20	0x0	LDI_OUTPUT_DATA: LCD_D22 pin output data 0 = LOW 1 = HIGH
19	0x0	LM1_OUTPUT_DATA_MASK: LCD_M1 pin output data mask 0 = MASKED 1 = NOTMASKED
18	0x0	LM1_OUTPUT_DATA: LCD_M1 pin output data 0 = LOW 1 = HIGH
17	0x0	LM0_OUTPUT_DATA_MASK: LCD_CS1_N pin output data mask 0 = MASKED 1 = NOTMASKED
16	0x0	LM0_OUTPUT_DATA: LCD_CS1_N pin output data 0 = LOW 1 = HIGH
11	0x0	LVP1_OUTPUT_DATA_MASK: LCD_D20 pin output data mask 0 = MASKED 1 = NOTMASKED
10	0x0	LVP1_OUTPUT_DATA: LCD_D20 pin output data 0 = LOW 1 = HIGH
9	0x0	LVP0_OUTPUT_DATA_MASK: LCD_DC1 pin output data mask 0 = MASKED 1 = NOTMASKED
8	0x0	LVP0_OUTPUT_DATA: LCD_DC1 pin output data 0 = LOW 1 = HIGH
5	0x0	LHP2_OUTPUT_DATA_MASK: LCD_D19 pin output data mask 0 = MASKED 1 = NOTMASKED

Bit	Reset	Description
4	0x0	LHP2_OUTPUT_DATA: LCD_D19 pin output data 0 = LOW 1 = HIGH
3	0x0	LHP1_OUTPUT_DATA_MASK: LCD_D18 pin output data mask 0 = MASKED 1 = NOTMASKED
2	0x0	LHP1_OUTPUT_DATA: LCD_D18 pin output data 0 = LOW 1 = HIGH
1	0x0	LHP0_OUTPUT_DATA_MASK: LCD_D21 pin output data mask 0 = MASKED 1 = NOTMASKED
0	0x0	LHP0_OUTPUT_DATA: LCD_D21 pin output data 0 = LOW 1 = HIGH

23.7.14 DC_COM_PIN_OUTPUT_DATA3_0

Pin Output Data 3

Pin Output data registers

Offset: 30dh | Read/Write: R/W | Reset: 0b000000000000

Bit	Reset	Description
11	0x0	LSDI_OUTPUT_DATA_MASK: LSDI pin output data mask 0 = MASKED 1 = NOTMASKED
10	0x0	LSDI_OUTPUT_DATA: LSDI pin output data 0 = LOW 1 = HIGH
9	0x0	LSPI_OUTPUT_DATA_MASK: LSPI pin output data mask 0 = MASKED 1 = NOTMASKED
8	0x0	LSPI_OUTPUT_DATA: LSPI pin output data 0 = LOW 1 = HIGH
7	0x0	LDC_OUTPUT_DATA_MASK: LDC pin output data mask 0 = MASKED 1 = NOTMASKED
6	0x0	LDC_OUTPUT_DATA: LDC pin output data 0 = LOW 1 = HIGH
5	0x0	LCSN_OUTPUT_DATA_MASK: LCSN pin output data mask 0 = MASKED 1 = NOTMASKED
4	0x0	LCSN_OUTPUT_DATA: LCSN pin output data 0 = LOW 1 = HIGH

Bit	Reset	Description
3	0x0	LSDA_OUTPUT_DATA_MASK: LSDA pin output data mask 0 = MASKED 1 = NOTMASKED
2	0x0	LSDA_OUTPUT_DATA: LSDA pin output data 0 = LOW 1 = HIGH
1	0x0	LSCK_OUTPUT_DATA_MASK: LSCK pin output data mask 0 = MASKED 1 = NOTMASKED
0	0x0	LSCK_OUTPUT_DATA: LSCK pin output data 0 = LOW 1 = HIGH

23.7.15 DC_COM_PIN_INPUT_DATA0_0

Pin Input Data 0

Pin Input Data registers (read-only)

Offset: 312h | Read/Write: RO | Reset: 0b000000000000000000

Bit	Reset	Description
17	0x0	LD17_INPUT_DATA: LD17 pin input data
16	0x0	LD16_INPUT_DATA: LD16 pin input data
15	0x0	LD15_INPUT_DATA: LD15 pin input data
14	0x0	LD14_INPUT_DATA: LD14 pin input data
13	0x0	LD13_INPUT_DATA: LD13 pin input data
12	0x0	LD12_INPUT_DATA: LD12 pin input data
11	0x0	LD11_INPUT_DATA: LD11 pin input data
10	0x0	LD10_INPUT_DATA: LD10 pin input data
9	0x0	LD9_INPUT_DATA: LD9 pin input data
8	0x0	LD8_INPUT_DATA: LD8 pin input data
7	0x0	LD7_INPUT_DATA: LD7 pin input data
6	0x0	LD6_INPUT_DATA: LD6 pin input data
5	0x0	LD5_INPUT_DATA: LD5 pin input data
4	0x0	LD4_INPUT_DATA: LD4 pin input data
3	0x0	LD3_INPUT_DATA: LD3 pin input data
2	0x0	LD2_INPUT_DATA: LD2 pin input data
1	0x0	LD1_INPUT_DATA: LD1 pin input data
0	0x0	LD0_INPUT_DATA: LD0 pin input data

23.7.16 DC_COM_PIN_INPUT_DATA1_0

Pin Input Data 1

Offset: 313h | Read/Write: RO | Reset: 0x0000000

Bit	Reset	Description
25	0x0	LSDI_INPUT_DATA: LSDI pin input data
24	0x0	LSPI_INPUT_DATA: LSPI pin input data
23	0x0	LDC_INPUT_DATA: LDC pin input data
22	0x0	LCSN_INPUT_DATA: LCSN pin input data
21	0x0	LSDA_INPUT_DATA: LSDA pin input data
20	0x0	LSCK_INPUT_DATA: LSCK pin input data
19	0x0	LPP_INPUT_DATA: LPP pin input data
18	0x0	LDI_INPUT_DATA: LDI pin input data
17	0x0	LM1_INPUT_DATA: LM1 pin input data
16	0x0	LM0_INPUT_DATA: LM0 pin input data
13	0x0	LVP1_INPUT_DATA: LVP1 pin input data
12	0x0	LVP0_INPUT_DATA: LVP0 pin input data
10	0x0	LHP2_INPUT_DATA: LHP2 pin input data
9	0x0	LHP1_INPUT_DATA: LHP1 pin input data
8	0x0	LHP0_INPUT_DATA: LHP0 pin input data
7	0x0	LHS_INPUT_DATA: LHS pin input data
6	0x0	LVS_INPUT_DATA: LVS pin input data
5	0x0	LSC1_INPUT_DATA: LSC1 pin input data
4	0x0	LSC0_INPUT_DATA: LSC0 pin input data
2	0x0	LPW2_INPUT_DATA: LPW2 pin input data
1	0x0	LPW1_INPUT_DATA: LPW1 pin input data
0	0x0	LPW0_INPUT_DATA: LPW0 pin input data

23.7.17 DC_COM_PIN_OUTPUT_SELECT0_0

3 bits are used to select output on each pin and they are defined as follows:

Pad Name	Pin Output Select							
	0	1	2	3	4	5	6	7
	Output Signal	Output Signal (*)	Output Signal	Output Signal	Output Signal	Output Signal	Output Signal	Output Signal
LD17	LD17	LD17 Out	LPD17	0	0	0	0	0
LD16	LD16	LD16 Out	LPD16	0	0	0	0	0
LD15	LD15	LD15 Out	LPD15	0	0	0	0	0
LD14	LD14	LD14 Out	LPD14	0	0	0	0	0
LD13	LD13	LD13 Out	LPD13	0	0	0	0	0
LD12	LD12	LD12 Out	LPD12	0	0	0	0	0
LD11	LD11	LD11 Out	LPD11	0	0	0	0	0
LD10	LD10	LD10 Out	LPD10	0	SD2	0	0	0
LD9	LD9	LD9 Out	LPD9	0	SD2_	0	0	0
LD8	LD8	LD8 Out	LPD8	0	STP	0	0	0
LD7	LD7	LD7 Out	LPD7	0	SDT	0	0	0
LD6	LD6	LD6 Out	LPD6	0	STH	0	0	0
LD5	LD5	LD5 Out	LPD5	0	SD1	0	0	0
LD4	LD4	LD4 Out	LPD4	0	SD1_	0	0	0
LD3	LD3	LD3 Out	LPD3	0	SD0	0	0	0
LD2	LD2	LD2 Out	LPD2	0	SD0_	0	0	0
LD1	LD1	LD1 Out	LPD1	0	SC	0	0	0
LD0	LD0	LD0 Out	LPD0	0	SC_	0	0	0
LPW0	PW0	LPW0 Out	PW1	PM0	PW2	MD0	LPD4	LSDA
LPW1	PW1	LPW1 Out	PW2	PM1	PW3	MD1	LPD8	PW4
LPW2	PW2	LPW2 Out	PW3	PM0	PW4	MD2	LPD5	PW1
LSC0	SC0	LSC0 Out	0	0	0	0	0	0
LSC1	SC1	LSC1 Out	DE	0	0	0	0	LSCK
LVS	Vsync	LVS Out	0	PM1	0	MD3	0	0
LHS	Hsync	LHS Out	0	PM0	0	MD2	0	0
LHP0	H Pulse 0	LHP0 Out	LD21	PM0	0	MD0	LPD7	0
LHP1	H Pulse 1	LHP1 Out	LD18	PM1	0	MD1	LPD0	0
LHP2	H Pulse 2	LHP2 Out	LD19	PM0	V Pulse 2	MD2	Hsync	0
LVP0	V Pulse 0	LVP0 Out	0	PM0	0	MD3	V Pulse1	LDC
LVP1	V Pulse 1	LVP1 Out	LD20	PM1	PW4	MD3	LPD6	0
LM0	M0	LM0 Out	H Pulse 0	PM0	V Pulse 2	MD0	DE	SCS_
LM1	M1	LM1 Out	LD21	PM1	V Pulse 3	MD1	LPD1	0
LDI	DI	LDI Out	LD22	PM0	SCS_	MD2	LPD2	0
LPP	PP	LPP Out	LD23	PM1	V Pulse 3	MD3	LPD3	0
LSCK	SCK	LSCK Out	0	PM0	0	MD0	LPD3	0
LSDA	SDA	LSDA Out	SCS_	PM1	0	MD1	LPD4	0
LCS_	SCS_	LCS_ Out	LD22	PM0	DE	MD2	LPD5	0
LDC	SDC	LDC Out	Vsync	PM1	V Pulse 1	MD3	LPD6	0
LSPI	SPI busy	LSPI Out	DE	PM0	DC Clk	MD0	0	0
LSDI	SDI	LSDI Out	0	PM1	0	MD1	0	0

Notes:

1. LD[23-0] contain pixel data for 1-pixel/1-clock parallel interface

2. LPD[17-0] contain pixel data for non 1-pixel/1-clock parallel interface

3. If output select is set to 1, then corresponding Pin Output Data register value is output (pin is used as general purpose output).

*4. If output select is set to 1 AND display is in DF2P1C18B (aka notebook) mode, then all output are overloaded. See table below. This doesn't apply to LSC0 (PCLK) which must be programmed to output-select==0.

5. For dual display, each display has its own instance of the above pad macro. Set up each display according to the panel interface and insure that pins are used by only one display. A set of registers outside of display, PIN_MUX_CTL_**, control which one out of 4 inputs, including display and displayb, go to the pads.

Dual Pixel Formats Table

Regarding note 4 above, in 2P1C18B mode, output-select==1 pin assignments are modified to accommodate two pixels' data per clock. Note that DF2P1C18B must be used with DIV2.

Internal Name	Ball Name	Single Channel 18-bpp	DF2P1C18B Dual Channel 18-bpp
ld[17:0]	lcd_db[17:0]	data[17:0]	odd[17:0]
ldc	lcd_dc0	--	even[17]
lcs	lcd_csn	--	even[16]
lsda	lcd_sda	--	even[15]
lsck	lcd_sck	--	even[14]
lpp	lcd_db[23]	--	even[13]
ldi	lcd_db[22]	--	even[12]
lm1	lcd_m1	--	even[11]
lm0	lcd_m0	--	even[10]
lvp1	lcd_db[20]	--	even[9]
lvp0	lcd_vp0	--	even[8]
lhp2	lcd_db[19]	--	even[7]
lhp1	lcd_db[18]	--	even[6]
lhp0	lcd_db[21]	--	even[5]
lhs	lcd_hsync	HS	HS
lvs	lcd_vsync	VS	VS
lpw2	lcd_pw2	--	even[2]
lpw1	lcd_pw1	--	even[1]
lpw0	lcd_pw0	--	even[0]
lspi	lcd_de	DE	DE
lsc1	lcd_wr_	--	even[3]
lsc0	lcd_pclk	CLK	CLK
lsdi	lcd_sdin	--	even[4]
dap3_din	dap3_din	--	--
dap3_dout	dap3_dout	--	--
dap3_fs	dap3_fs	--	--
dap3_sclk	dap3_sclk	--	--

Pin Output Select 0

Offset: 314h | Read/Write: R/W | Reset: 0b000x000x000x000x000x000x000x000

Bit	Reset	Description
30:28	0x0	LD7_OUTPUT_SELECT: LD7 pin output select
26:24	0x0	LD6_OUTPUT_SELECT: LD6 pin output select
22:20	0x0	LD5_OUTPUT_SELECT: LD5 pin output select
18:16	0x0	LD4_OUTPUT_SELECT: LD4 pin output select
14:12	0x0	LD3_OUTPUT_SELECT: LD3 pin output select
10:8	0x0	LD2_OUTPUT_SELECT: LD2 pin output select
6:4	0x0	LD1_OUTPUT_SELECT: LD1 pin output select
2:0	0x0	LD0_OUTPUT_SELECT: LD0 pin output select

23.7.18 DC_COM_PIN_OUTPUT_SELECT1_0

Pin Output Select 1

Offset: 315h | Read/Write: R/W | Reset: 0b000x000x000x000x000x000x000x000

Bit	Reset	Description
30:28	0x0	LD15_OUTPUT_SELECT: LD15 pin output select
26:24	0x0	LD14_OUTPUT_SELECT: LD14 pin output select
22:20	0x0	LD13_OUTPUT_SELECT: LD13 pin output select
18:16	0x0	LD12_OUTPUT_SELECT: LD12 pin output select
14:12	0x0	LD11_OUTPUT_SELECT: LD11 pin output select
10:8	0x0	LD10_OUTPUT_SELECT: LD10 pin output select
6:4	0x0	LD9_OUTPUT_SELECT: LD9 pin output select
2:0	0x0	LD8_OUTPUT_SELECT: LD8 pin output select

23.7.19 DC_COM_PIN_OUTPUT_SELECT2_0

Pin Output Select 2

Offset: 316h | Read/Write: R/W | Reset: 0b000x000

Bit	Reset	Description
6:4	0x0	LD17_OUTPUT_SELECT: LD17 pin output select
2:0	0x0	LD16_OUTPUT_SELECT: LD16 pin output select

23.7.20 DC_COM_PIN_OUTPUT_SELECT3_0

Pin Output Select 3

Offset: 317h | Read/Write: R/W | Reset: 0b000x000x000x000xxxxx000x000x000

Bit	Reset	Description
30:28	0x0	LHS_OUTPUT_SELECT: LCD_HSYNC pin output select
26:24	0x0	LVS_OUTPUT_SELECT: LCD_VSYNC pin output select
22:20	0x0	LSC1_OUTPUT_SELECT: LCD_WR_N pin output select
18:16	0x0	LSC0_OUTPUT_SELECT: LCD_PCLK pin output select
10:8	0x0	LPW2_OUTPUT_SELECT: LCD_PWR2 pin output select
6:4	0x0	LPW1_OUTPUT_SELECT: LCD_PWR1 pin output select
2:0	0x0	LPW0_OUTPUT_SELECT: LCD_PWR0 pin output select

23.7.21 DC_COM_PIN_OUTPUT_SELECT4_0

Pin Output Select 4

Offset: 318h | Read/Write: R/W | Reset: 0b000x000xxxxx000x000x000

Bit	Reset	Description
22:20	0x0	LVP1_OUTPUT_SELECT: LCD_D20 pin output select
18:16	0x0	LVP0_OUTPUT_SELECT: LCD_DC1 pin output select
10:8	0x0	LHP2_OUTPUT_SELECT: LCD_D19 pin output select
6:4	0x0	LHP1_OUTPUT_SELECT: LCD_D18 pin output select
2:0	0x0	LHP0_OUTPUT_SELECT: LCD_D21 pin output select

23.7.22 DC_COM_PIN_OUTPUT_SELECT5_0

Pin Output Select 5

Offset: 319h | Read/Write: R/W | Reset: 0b000x000x000x000

Bit	Reset	Description
14:12	0x0	LPP_OUTPUT_SELECT: LCD_D23 pin output select
10:8	0x0	LDI_OUTPUT_SELECT: LCD_D22 pin output select
6:4	0x0	LM1_OUTPUT_SELECT: LCD_M1 pin output select
2:0	0x0	LM0_OUTPUT_SELECT: LCD_CS1_N pin output select

23.7.23 DC_COM_PIN_OUTPUT_SELECT6_0

Pin Output Select 6

Offset: 31ah | Read/Write: R/W | Reset: 0b000x000x000x000x000x000

Bit	Reset	Description
22:20	0x0	LSDI_OUTPUT_SELECT: LSDI pin output select
18:16	0x0	LSPI_OUTPUT_SELECT: LSPI pin output select
14:12	0x0	LDC_OUTPUT_SELECT: LDC pin output select
10:8	0x0	LCSN_OUTPUT_SELECT: LCSN pin output select
6:4	0x0	LSDA_OUTPUT_SELECT: LSDA pin output select
2:0	0x0	LSCK_OUTPUT_SELECT: LSCK pin output select

23.7.24 DC_COM_PIN_MISC_CONTROL_0

Pin Miscellaneous Control

Offset: 31bh | Read/Write: R/W | Reset: 0b0

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

23.7.25 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: 31ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:18	X	PM0_PERIOD: PM0 Period (4, 8, ... , 256 clock cycles)
9:4	X	PM0_CLOCK_DIVIDER: PM0 Clock Divider (1 to 16384)

Bit	Reset	Description
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 is running only when 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.

23.7.26 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: 31dh | Read/Write: R/W | Reset: 0bxxxxxxxxx

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. This must not be larger than the period.

23.7.27 DC_COM_PM1_CONTROL_0

PM1 signal Control

PM1 signal is programmable pulse width modulation signal that can be output on several pins.

The control register should be initialized once before PM1 is enabled.

The actual period (in clock cycles) is given by:

PERIOD = (1 + PM1_CLOCK_DIVIDER) * ((PM1_PERIOD + 1) * 4)

Offset: 31eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:18	X	PM1_PERIOD: PM1 Period (4, 8, ... , 256 clock cycles)
9:4	X	PM1_CLOCK_DIVIDER: PM1 Clock Divider (1 to 16384)
1:0	X	PM1_CLOCK_SELECT: PM1 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 is running only when 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.

23.7.28 DC_COM_PM1_DUTY_CYCLE_0

PM1 Duty Cycle

A counter repeatedly counts up from 0 to ((PM1_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 ((PM1_PERIOD << 2) + 3).

Offset: 31fh | Read/Write: R/W | Reset: 0bxxxxxxxxx

Bit	Reset	Description
8:0	X	PM1_DUTY_CYCLE: PM1 Duty Cycle from 1/P to P/P pulse high time where P is the period. This must not be larger than the period.

23.7.29 DC_COM_SPI_CONTROL_0

SPI Control

Class: Serial Peripheral Interface (SPI)

SPI interface is a 3-pin or 4-pin serial interface which is typically used to program registers in display device. However, for display with built-in frame buffer, it can also be used to write pixel data to the built-in frame buffer. Currently only write cycles are supported.

LCD SPI interface signal consists of:

- SPI Clock (SCK) which can be output on LCD_SCK pin.
- SPI Data (SDA) which can be output on LCD_SDOUT pin.
- Optional SPI Data/Command (SDC) which can be output on LCD_DC0 pin.
- Main-Display SPI Chip Select (Main SCS_) signal which can be output on LCS_ pin
- Sub-Display SPI Chip Select (Sub SCS_) signal which can be output on LCD_CS1_N or LCD_D22 or LCD_SDOUT pins - this is only used only if there is a sub display

Theoretically if two LCD panels (main and sub displays) are connected, then SPI interface can be connected to both LCD panels and shared between the two panels.

In this case two Chip Select pins are required and LCS_ should be connected to the main display and other pin has to be selected to output the sub display SCS_.

Internally two SPI chip select signals (lmscs_ and lsscs_ for main and sub display SPI chip select) are generated.

When SPI is enabled, there are three possible SPI transactions:

- SPI write can be triggered by host by writing to HSPI_CS_DC register
- SPI write can be triggered to send pixel data from frame buffer
- SPI write can be triggered to send Initialization Sequence (IS). Frame initialization sequence is sent one line prior to active display line.

For LCD SPI, pixel data can only be sent to either Main-Display or Sub-Display but not to both but host SPI and IS SPI can be sent to both Main and Sub displays simultaneously.

In case a. and c. the SPI root clock is derived from output of shift clock divider which is then further divided by SPI_CLK_DIVIDER. For both cases, the SPI number of bits/cycle is determined by SPI_BITS_PER_CYCLE and the SPI data direction is determined by SPI_DATA_DIRECTION.

In case b. the SPI root clock is derived from output of shift clock divider with no further division. For this case, the SPI number of bits/cycle is determined by correct programming of pixel clock divider.

If case a. is enabled at the same time with case b. and c. then host SPI triggers is delayed to the beginning of horizontal display active time of a line that does not have IS SPI or LCD SPI cycles. This means that if all cases are enabled, the vertical blank time must be at least 2 lines otherwise host SPI cycles cannot be executed.

Offset: 320h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25:24	X	SPI_STATUS_ENABLE: SPI Status Enable 00= SPI status disable 01= SPI status enabled for host SPI only 10= SPI status enabled for IS and LCD SPI only 11= SPI status enabled for all SPI cycles SPI status is reflected in SPI_BUSY bit and can generate interrupt. SPI status indicates when SPI module is busy (SPI write cycles are in progress) so its falling edge should be used to generate interrupt. SPI status can also be output on LCD_DE pin. When Host SPI is triggered, the SPI busy is asserted within three display clock cycles after the end of the host write cycle. SPI status is disabled when SPI_ENABLE bit is disabled.
20:16	X	SPI_CLK_DIVIDER: SPI Clock Divider (1 to 32). This clock divider is used only if SPI is enabled for host writes (Host SPI) or for sending initialization sequence (IS SPI). Programmed value is 1 less than the desired (actual) clock divider value. This parameter is forced to 0 (clock divide by 1) for LCD SPI.
7:4	X	SPI_BITS_PER_CYCLE: SPI Bits per Cycle. This is valid for Host and IS SPI only. This parameter determines the number of bits/cycle when SPI is used for host write or for sending initialization sequence. If SPI is used for sending pixel data to the display then pixel clock divider determines the SPI bits/cycle. SPI8DC is 8-bit SPI plus data/command bit SPI16DC is 16-bit SPI plus data/command bit SPI16SB is 16-bit SPI plus an 8-bit start byte preceding the 16-bit data. 0 = SPI8 1 = SPI8DC 2 = SPI12 3 = SPI16 4 = SPI16DC 5 = SPI16SB 6 = SPI18 7 = SPI24
3	X	SPI_DATA_DIRECTION: SPI Data Direction. This is valid for Host SPI and for sending initialization sequence (IS SPI) only. Note that data direction does not affect the start byte direction (always msb to lsb) and position (always first 8-bit of serial data) for SPI16SB mode. 0 = MSB2LSB 1 = LSB2MSB
1:0	X	SPI_SERIAL_CLK_CONTROL: SPI Serial Clock Control 0= SCK rising edge is active edge 1-clock chip select and no SCK clock edge to latch chip select 1= SCK rising edge is active edge 2-clock chip select with SCK rising clock edge to latch it 2= SCK falling edge is active edge 1-clock chip select and no SCK clock edge to latch chip select 3= SCK falling edge is active edge 2-clock chip select with SCK falling clock edge to latch it This is valid for Host, IS, and LCD SPI

23.7.30 DC_COM_SPI_START_BYTE_0

SPI Start Byte

SPI Start Bytes are used only for SPI16SB mode (start byte plus 16-bit data). Data direction does not affect the start byte direction (always msb to lsb) and position (always the first 8 bits of serial data).

Offset: 321h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:8	X	SPI_COMMAND_START_BYTE: SPI Command Start Byte This is valid for Host, IS, and LCD SPI.
7:0	X	SPI_DATA_START_BYTE: SPI Data Start Byte. This is valid for Host, IS, and LCD SPI.

23.7.31 DC_COM_HSPI_WRITE_DATA_AB_0

Host SPI Write Data A & B

These registers are used to send write data for Host SPI write cycles (HSPI_ENABLE = 1).

For the tables below, these abbreviations of the registers are used:

- HSPIWDA = HSPI_WRITE_DATA_A
- HSPIWDB = HSPI_WRITE_DATA_B
- HSPIWDC = HSPI_WRITE_DATA_C
- HSPIWDD = HSPI_WRITE_DATA_D
- HSPIMCS = HSPI_MAIN_CS
- HSPISCS = HSPI_SUB_CS
- HSPIDC = HSPI_DC

For 8-bit SPI, up to 4 write cycles can be performed (SPI_BITS_PER_CYCLE equals SPI8)

Write cycle	0	1	2	3
Data	HSPIWDA[7-0]	HSPIWDA[15-8]	HSPIWDB[7-0]	HSPIWDB[15-8]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]	HSPIMCS[2]	HSPIMCS[3]
Sub SCS_	HSPISCS[0]	HSPISCS[1]	HSPISCS[2]	HSPICSC[3]
SDC	HSPIDC[0]	HSPIDC[1]	HSPIDC[2]	HSPIDC[3]

For SPI8DC mode, a 9-bit SPI, the first data bit sent comes from HSPI_DC register and the other eight bits come as shown in the table below. Up to 4 write cycles can be performed.

Write cycle	0	1
Data	HSPIDC[0], HSPIWDA[7-0]	HSPIDC[1], HSPIWDA[15-8]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]
Sub SCS_	HSPISCS[0]	HSPISCS[1]
SDC	HSPIDC[0]	HSPIDC[1]
Write cycle	**2**	**3**
Data	HSPIDC[2], HSPIWDB[7-0]	HSPIDC[3], HSPIWDB[15-8]
Main SCS_	HSPIMCS[2]	HSPIMCS[3]
Sub SCS_	HSPISCS[2]	HSPICSC[3]
SDC	HSPIDC[2]	HSPIDC[3]

For 12-bit SPI, up to 4 write cycles can be performed (SPI12)

Write cycle	0	1	2	3
Data	HSPIWDA[11-0]	HSPIWDB[11-0]	HSPIWDC[11-0]	HSPIWDD[11-0]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]	HSPIMCS[2]	HSPIMCS[3]
Sub SCS_	HSPISCS[0]	HSPISCS[1]	HSPISCS[2]	HSPISCS[3]
SDC	HSPIDC[0]	HSPIDC[1]	HSPIDC[2]	HSPIDC[3]

For 16-bit SPI, up to 4 write cycles can be performed (SPI16, SPI16SB). Note for SPI16SB mode, the first write cycle will be the start byte, followed by write cycle 0, as shown in the table below.

Write cycle	0	1	2	3
Data	HSPIWDA[15-0]	HSPIWDB[15-0]	HSPIWDC[15-0]	HSPIWDD[15-0]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]	HSPIMCS[2]	HSPIMCS[3]
Sub SCS_	HSPISCS[0]	HSPISCS[1]	HSPISCS[2]	HSPISCS[3]
SDC	HSPIDC[0]	HSPIDC[1]	HSPIDC[2]	HSPIDC[3]

For SPI16DC mode, a 17-bit SPI, the first data bit sent comes from HSPI_DC register and the other sixteen bits come as shown in the table below. Up to 4 write cycles can be performed.

Write cycle	0	1
Data	HSPIDC[0], HSPIWDA[15-0]	HSPIDC[1], HSPIWDB[15-0]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]
Sub SCS_	HSPISCS[0]	HSPISCS[1]
SDC	HSPIDC[0]	HSPIDC[1]
Write cycle	**2**	**3**
Data	HSPIDC[2], HSPIWDC[15-0]	HSPIDC[3], HSPIWDD[15-0]
Main SCS_	HSPIMCS[2]	HSPIMCS[3]
Sub SCS_	HSPISCS[2]	HSPISCS[3]
SDC	HSPIDC[2]	HSPIDC[3]

For 18-bit SPI, up to 2 write cycles can be performed (SPI18)

Write cycle	0	1
Data	HSPIWDB[1-0], HSPIWDA[15-0]	HSPIWDC[1-0], HSPIWDD[15-0]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]
Sub SCS_	HSPISCS[0]	HSPISCS[1]
SDC	HSPIDC[0]	HSPIDC[1]

For 24-bit SPI, up to 2 write cycles can be performed (SPI24)

Write cycle	0	1
Data	HSPIWDB[7-0], HSPIWDA[15-0]	HSPIWDC[7-0], HSPIWDD[15-0]
Main SCS_	HSPIMCS[0]	HSPIMCS[1]
Sub SCS_	HSPISCS[0]	HSPISCS[1]
SDC	HSPIDC[0]	HSPIDC[1]

Offset: 322h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	HSPI_WRITE_DATA_B: Host SPI Write Data B bits 15-0
15:0	X	HSPI_WRITE_DATA_A: Host SPI Write Data A bits 15-0

23.7.32 DC_COM_HSPI_WRITE_DATA_CD_0

Host SPI Write Data C & D

Offset: 323h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	HSPI_WRITE_DATA_D: Host SPI Write Data D bits 15-0
15:0	X	HSPI_WRITE_DATA_C: Host SPI Write Data C bits 15-0

23.7.33 DC_COM_HSPI_CS_DC_0

Host SPI Chip Select and Data/Command

A write to this register will trigger the Host SPI write cycle if SPI_ENABLE and HSPI_ENABLE are both enabled. Writing to this register with HSPI_RAISE enabled will cause the raise vector to be returned after all the host SPI cycles are completed. Up to four host SPI cycles can be executed with a single trigger.

This register should not be written if previous host SPI write cycle is in progress. HSPI_MAIN_CS and HSPI_SUB_CS controls the main and sub display chip selects and therefore also determine how many SPI write cycles to main and sub displays and the write data position.

HSPI_MAIN_CS or HSPI_SUB_CS should be programmed to have at least one valid cycle when programming HSPI_CS_DC register.

Offset: 324h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx0

Bit	Reset	Description
28:24	X	HSPI_RAISE_VECTOR: Host SPI Raise Vector This raise vector is returned after all the triggered host SPI cycles are executed.
19:16	X	HSPI_RAISE_CHANNEL_ID: Win G Channel ID
15:12	X	HSPI_SUB_CS: Host SPI Sub display Chip Select (Sub SCS_) 0= Sub display not selected (Sub SCS_=1) 1= Sub display selected (Sub SCS_=0) This is valid for Host SPI only. Each bit of this parameter corresponds to the four possible host SPI cycles.
11:8	X	HSPI_MAIN_CS: Host SPI Main display Chip Select (Main SCS_) 0= Main display not selected (Main SCS_=1) 1= Main display selected (Main SCS_=0) This is valid for Host SPI only. Each bit of this parameter corresponds to the four possible host SPI cycles.
7:4	X	HSPI_DC: Host SPI Data/Command_ (SDC) 0= Command cycle (SDC=0) 1= Data cycle (SDC=1) This is valid for Host SPI only. Each bit of this parameter corresponds to the four possible host SPI cycles.
0	0x0	HSPI_RAISE: Host SPI Raise. Raise vector will be returned at the end of the host SPI write cycles 0 = DISABLE 1 = ENABLE

23.7.34 DC_COM_SCRATCH_REGISTER_A_0

Scratch Register A

Class: Software Scratch Registers

Offset: 325h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SCRATCH_REGISTER_A: Scratch Register A

23.7.35 DC_COM_SCRATCH_REGISTER_B_0

Scratch Register B

Offset: 326h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SCRATCH_REGISTER_B: Scratch Register B

23.7.36 DC_COM_GPIO_CTRL_0

Display GPIO control, including debounce control

Offset: 327h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2	0x0	GPIO_2_DEBOUNCE_ENABLE: maps to display pin LCD_PWR2 0 = DISABLE 1 = ENABLE
1	0x0	GPIO_1_DEBOUNCE_ENABLE: maps to display pin LCD_PWR1 0 = DISABLE 1 = ENABLE
0	0x0	GPIO_0_DEBOUNCE_ENABLE: maps to display pin LCD_PWR0 0 = DISABLE 1 = ENABLE

23.7.37 DC_COM_GPIO_DEBOUNCE_COUNTER_0

Display GPIO Debounce Counter

Provides mark_en to debounce logic.

MARKER

The MARKER value sets the interval at which the debounce finite state machines associated with each GPIO input pin evaluate input transitions. An input transition must be stable for at least 4 consecutive MARKER ticks before it is propagated through the debounce circuit.

MARKER is in display clock tick units. The actual 'gravity delay' of a debounce FSM is 4 * Tmarker, where Tmarker = MARKER * (APB clock duration). For a 100 MHz display clock, a MARKER setting of 10,000,000 results in a gravity delay of 4x100 ms.

Offset: 328h | Read/Write: R/W | Reset: 0b00000000000000000000010000000000

Bit	Reset	Description
31:0	0x400	DEBOUNCE_COUNTER

23.7.38 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: 329h | Read/Write: RO | Reset: 0x00000000

Bit	Reset	Description
31:0	0x0	CRC_CHECKSUM_LATCHED: CRC Checksum latched

23.8 Display DISP Registers

These registers control DISP display only; not including the DISP display window parameters.

23.8.1 DC_DISP_DISP_SIGNAL_OPTIONS0_0

Display Signal Options 0

Offset: 400h | Read/Write: R/W | Reset: 0b0x0xxx000x0xxx0x0x0

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

23.8.2 DC_DISP_DISP_SIGNAL_OPTIONS1_0

Display Signal Options 1

Offset: 401h | Read/Write: R/W | Reset: 0b0x0

Bit	Reset	Description
18	0x0	PP_ENABLE: PP Enable 0 = DISABLE 1 = ENABLE
16	0x0	DI_ENABLE: DI Enable 0 = DISABLE 1 = ENABLE

23.8.3 DC_DISP_DISP_WIN_OPTIONS_0

Display Window Options

Offset: 402h | Read/Write: R/W | Reset: 0b000xxxxxxxxxx0

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
28	0x0	TVO_ENABLE: TVO Enable. Steps to start displaying on TV (The order of the first 3 steps can freely change): -- Program and enable TVO module -- Program DISPLAY_CTRL_MODE to NC_DISPLAY -- Program the ASSEMBLY shadow copy of this register field with ENABLE -- Program GENERAL_ACT_REQ to activate the shadow 0 = DISABLE 1 = ENABLE
16	0x0	CURSOR_ENABLE: Cursor Enable 0 = DISABLE 1 = ENABLE

23.8.4 DC_DISP_MEM_HIGH_PRIORITY_0

Memory High Priority request control

Display Memory High Priority Threshold

This controls high priority request for memory read access for each display window and for cursor. High priority request threshold should be increased in scenarios where memory access latency is high.

> **Note:** High Priority Threshold is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 403h | Read/Write: R/W | Reset: 0b00000000000x000000000000000

Bit	Reset	Description
26:24	0x0	CSR_DISPLAYHC2MC_HPTH: Cursor Memory High Priority enable 0= memory access for cursor is normal priority 1= memory access for cursor is high priority
23:16	0x0	CBR_DISPLAY0C2MC_HPTH: Window C Memory High Priority threshold Memory access for this window is high priority if the number of filled entries in the return data fifo is less than or equal to this value. Setting this parameter to 0 disables high priority memory request.
14:8	0x0	CBR_DISPLAYB2MC_HPTH: Window B Memory High Priority threshold Memory access for this window is high priority if the number of filled entries in the return data fifo is less than or equal to this value. Setting this parameter to 0 disables high priority memory request. This register is used for both window B0 and B1
7:0	0x0	CBR_DISPLAY0A2MC_HPTH: Window A Memory High Priority threshold Memory access for this window is high priority if the number of filled entries in the return data fifo is less than or equal to this value. Setting this parameter to 0 disables high priority memory request.

23.8.5 DC_DISP_MEM_HIGH_PRIORITY_TIMER_0

Memory High Priority request control

Display Memory High Priority Timer

The high-priority assertion can be delayed by a number of memory clock cycles indicated by the timer. This creates an hysteresis effect, avoiding setting the high-priority for very short periods of time, which may or may not be desirable.

> **Note:** High Priority Timer is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 404h | Read/Write: R/W | Reset: 0b000000xx000000xx000000xx000000

Bit	Reset	Description
29:24	0x0	CSR_DISPLAYHC2MC_HPTM: Cursor Memory High Priority timer
21:16	0x0	CBR_DISPLAY0C2MC_HPTM: Window C Memory High Priority timer
13:8	0x0	CBR_DISPLAYB2MC_HPTM: Window B Memory High Priority timer This register is used for both window B0 and B1
5:0	0x0	CBR_DISPLAY0A2MC_HPTM: Window A Memory High Priority timer

23.8.6 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 > 11.

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

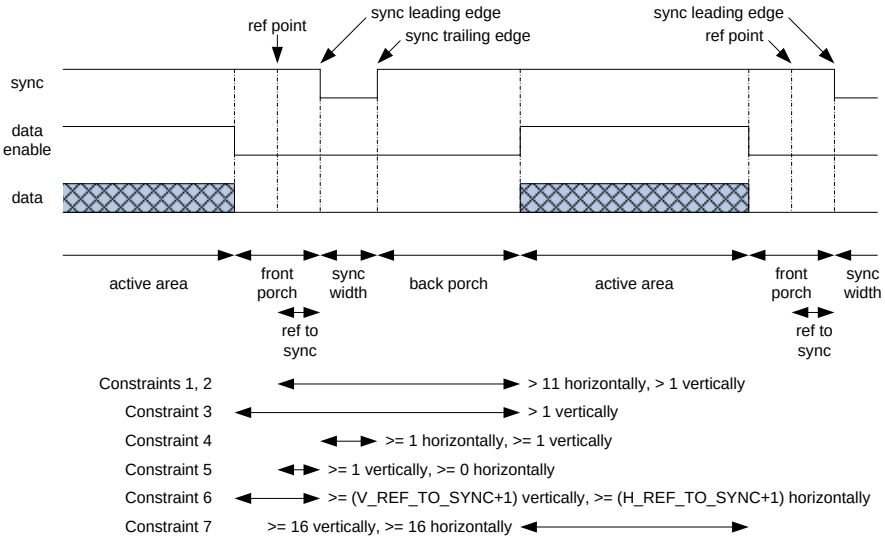
Only the back porch can be a negative value, though it is typically positive. Note that the back porch is defined as the distance between the trailing edge of the sync and the beginning of the active area. Some LCD specifications define the back porch from the leading edge of the sync rather than the trailing edge. Front porch is defined as the distance between the end of the active area and the leading edge of the sync. Vertical timing is in terms of lines and horizontal timing is in terms of pixels.

Sync polarity is in terms of the sync width; active low means the sync width pulse will be low, as in the diagram. Pixel clock polarity is in terms of when data will transition; an active low pixel clock means that the data changes on the falling edge of the clock, to be latched on the rising edge. Data enable (also called display enable or DE) polarity selects the level of the signal during the active area of display. The diagram has DE active high.

Timing Diagram

- This diagram applies to both vertical and horizontal timing
- Back porch is the only parameter that can be negative

Figure 54 Display Timing Options Timing Diagram



This register specifies display timing options for HSYNC and VSYNC

Offset: 405h | Read/Write: R/W | Reset: 0b0000000000000

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.

23.8.7 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. In the future, we may want to add a horizontal position offset so that VSYNC can occur after HSYNC.

Offset: 406h | Read/Write: R/W | Reset: 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)

23.8.8 DC_DISP_SYNC_WIDTH_0

H/V SYNC Pulse Width

This register specifies the width of HSYNC and VSYNC pulses. Check the comment for REF_TO_SYNC for programming restrictions.

Offset: 407h | Read/Write: R/W | Reset: 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)

23.8.9 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 2's complement value and negative value indicates that H/V SYNC overlaps with the corresponding display active area. Check the comment for REF_TO_SYNC for programming restrictions.

Offset: 408h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	V_BACK_PORCH: V back porch
12:0	X	H_BACK_PORCH: H back porch

23.8.10 DC_DISP_DISP_ACTIVE_0

H/V Display Active width

This register specifies the width of H/V display active area. Check the comment for REF_TO_SYNC for programming restrictions.

Offset: 409h | Read/Write: R/W | Reset: 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)

23.8.11 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.

Design Note: 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 comment for REF_TO_SYNC for programming restrictions.

Offset: 40ah | Read/Write: R/W | Reset: 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)

23.8.12 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 the 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 Enable field is set to Start A position then no pulse is generated.

In the 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: 40bh | Read/Write: R/W | Reset: 0bxxxxxxxxx

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

Bit	Reset	Description
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

23.8.13 DC_DISP_H_PULSE0_POSITION_A_0

H Pulse 0 Position A

Offset: 40ch | Read/Write: R/W | Reset: 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)

23.8.14 DC_DISP_H_PULSE0_POSITION_B_0

H Pulse 0 Position B

Offset: 40dh | Read/Write: R/W | Reset: 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)

23.8.15 DC_DISP_H_PULSE0_POSITION_C_0

H Pulse 0 Position C

Offset: 40eh | Read/Write: R/W | Reset: 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)

23.8.16 DC_DISP_H_PULSE0_POSITION_D_0

H Pulse 0 Position D

Offset: 40fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	H_PULSE0_END_D: H Pulse 0 End D (minimum -=PS_=-H_PULSE0_START_D+1)

Bit	Reset	Description
12:0	X	H_PULSE0_START_D: H Pulse 0 Start D (minimum -=PS_=-H_PULSE0_END_C+1)

23.8.17 DC_DISP_H_PULSE1_CONTROL_0

H Pulse 1 Control

Horizontal pulse 1 is programmable pulse that repeats every line.

In the 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 Enable field is set to Start A position then no pulse is generated.

In the 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: 410h | Read/Write: R/W | Reset: 0bxxxxxxxxx

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

Bit	Reset	Description
3	X	H_PULSE1_MODE: H Pulse 1 Mode 0= Normal mode 1= Single-clock mode 0 = NORMAL 1 = ONE_CLOCK

23.8.18 DC_DISP_H_PULSE1_POSITION_A_0

H Pulse 1 Position A

Offset: 411h | Read/Write: R/W | Reset: 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)

23.8.19 DC_DISP_H_PULSE1_POSITION_B_0

H Pulse 1 Position B

Offset: 412h | Read/Write: R/W | Reset: 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)

23.8.20 DC_DISP_H_PULSE1_POSITION_C_0

H Pulse 1 Position C

Offset: 413h | Read/Write: R/W | Reset: 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)

23.8.21 DC_DISP_H_PULSE1_POSITION_D_0

H Pulse 1 Position D

Offset: 414h | Read/Write: R/W | Reset: 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)

23.8.22 DC_DISP_H_PULSE2_CONTROL_0

H Pulse 2 Control

Horizontal pulse 2 is programmable pulse that repeats every line.

In the 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 Enable field is set to Start A position then no pulse is generated.

In the 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: 415h | Read/Write: R/W | Reset: 0bxxxxxxxxx

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

23.8.23 DC_DISP_H_PULSE2_POSITION_A_0

H Pulse 2 Position A

Offset: 416h | Read/Write: R/W | Reset: 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)

23.8.24 DC_DISP_H_PULSE2_POSITION_B_0

H Pulse 2 position B

Offset: 417h | Read/Write: R/W | Reset: 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)

23.8.25 DC_DISP_H_PULSE2_POSITION_C_0

H Pulse 2 Position C

Offset: 418h | Read/Write: R/W | Reset: 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)

23.8.26 DC_DISP_H_PULSE2_POSITION_D_0

H Pulse 2 Position D

Offset: 419h | Read/Write: R/W | Reset: 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)

23.8.27 DC_DISP_V_PULSE0_CONTROL_0

V Pulse 0 Control

Vertical pulse 0 is 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 Enable field is set to 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: 41ah | Read/Write: R/W | Reset: 0b0000000000000xxxxxxxxxxxx

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 0 = HIGH 1 = LOW

23.8.28 DC_DISP_V_PULSE0_POSITION_A_0

V Pulse 0 Position A

Offset: 41bh | Read/Write: R/W | Reset: 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)

23.8.29 DC_DISP_V_PULSE0_POSITION_B_0

V Pulse 0 Position B

Offset: 41ch | Read/Write: R/W | Reset: 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)

23.8.30 DC_DISP_V_PULSE0_POSITION_C_0

V Pulse 0 Position C

Offset: 41dh | Read/Write: R/W | Reset: 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)

23.8.31 DC_DISP_V_PULSE1_CONTROL_0

V pulse 1 Control

Vertical pulse 1 is 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 Enable field is set to 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: 41eh | Read/Write: R/W | Reset: 0b0000000000000xxxxxxxxxxxx

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.

Bit	Reset	Description
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
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

23.8.32 DC_DISP_V_PULSE1_POSITION_A_0

V Pulse 1 Position A

Offset: 41fh | Read/Write: R/W | Reset: 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)

23.8.33 DC_DISP_V_PULSE1_POSITION_B_0

V Pulse 1 Position B

Offset: 420h | Read/Write: R/W | Reset: 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)

23.8.34 DC_DISP_V_PULSE1_POSITION_C_0

V Pulse 1 Position C

Offset: 421h | Read/Write: R/W | Reset: 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)

23.8.35 DC_DISP_V_PULSE2_CONTROL_0

V pulse 2 Control

Vertical pulse 2 is 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 Enable field is set to 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: 422h | Read/Write: R/W | Reset: 0b0000000000000xxxxxxxxxxxx

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 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	X	V_PULSE2_POLARITY: V pulse 2 Polarity 0 = HIGH 1 = LOW

23.8.36 DC_DISP_V_PULSE2_POSITION_A_0

V Pulse 2 Position A

Offset: 423h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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)

23.8.37 DC_DISP_V_PULSE3_CONTROL_0

V pulse 3 Control

Vertical pulse 3 is 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 Enable field is set to 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: 424h | Read/Write: R/W | Reset: 0b0000000000000xxxxxxxxxxx0

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

23.8.38 DC_DISP_V_PULSE3_POSITION_A_0

V Pulse 3 Position A

Offset: 425h | Read/Write: R/W | Reset: 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)

23.8.39 DC_DISP_M0_CONTROL_0

M0 Control

Display M0 signal

M0 signal can be generated either using a line (horizontal) or a frame (vertical) clock and it can be horizontally positioned with respect to H reference point. This signal is typically output on LCD_CS1_N pin.

This register specifies options for M0 signal.

Offset: 426h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	M0_H_POSITION: M0 Horizontal Position. This parameter specifies the position where M0 can toggle with respect to H reference point.
12:8	X	M0_PERIOD: M0 Period. This should be program to the half of the desired M0 period (in lines) minus 1.
7	X	M0_POLARITY: M0 Polarity. Polarity adjustment is applied last after phase control is applied. 0 = HIGH 1 = LOW
6	X	M0_PHASE_RESET: M0 Phase Reset. This bit is effective only when M0 is not free running. 0= frequency (phase) counter is not reset 1= frequency (phase) counter is reset at beginning of vertical active display if phase control is set to VACTIVE_RESTART or at beginning of frame if phase control is set to FRAME_INVERT 0 = DISABLE 1 = ENABLE
5:4	X	M0_PHASE_CONTROL: M0 Phase Control 00= free-running 01= reserved 10= reset at beginning of vertical active display 11= invert at beginning of frame This should be set to free-running if frame clock is used. 0 = FREE_RUN 2 = VACTIVE_RESTART 3 = FRAME_INVERT
1:0	X	M0_CLOCK_SELECT: M0 Clock Select 00= pixel clock (for diagnostic) 01= reserved 10= line clock 11= frame clock 0 = PCLK 2 = LCLK 3 = FCLK

23.8.40 DC_DISP_M1_CONTROL_0

M1 Control

Display M1 signal. M1 signal can be generated either using a line (horizontal) or a frame (vertical) clock and it can be horizontally positioned with respect to H reference point. This signal is typically output on LCD_M1 pin. This register specifies options for M1 signal.

Offset: 427h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	M1_H_POSITION: M1 Horizontal Position. This parameter specifies the position where M0 can toggle with respect to H reference point.
12:8	X	M1_PERIOD: M1 Period. This should be program to the half of the desired M1 period (in lines) minus 1.
7	X	M1_POLARITY: M1 Polarity. Polarity adjustment is applied last after phase control is applied. 0 = HIGH 1 = LOW

Bit	Reset	Description
6	X	M1_PHASE_RESET: M1 Phase Reset. This bit is effective only when M1 is not free running. 0= frequency (phase) counter is not reset 1= frequency (phase) counter is reset at beginning of vertical active display if phase control is set to VACTIVE_RESTART or at beginning of frame if phase control is set to FRAME_INVERT 0 = DISABLE 1 = ENABLE
5:4	X	M1_PHASE_CONTROL: M1 Phase Control 00= free-running 01= reserved 10= reset at beginning of vertical active display 11= invert at beginning of frame This should be set to free-running if frame clock is used. 0 = FREE_RUN 2 = VACTIVE_RESTART 3 = FRAME_INVERT
1:0	X	M1_CLOCK_SELECT: M1 Clock Select 00= pixel clock (for diagnostic) 01= synchronous to M0 provided that M0 is generated using line clock. This will not work if M0 is not generated using line clock. In this case, M1 is controlled by -=PS_=-M0_PHASE_RESET and -=PS_=-M0_PERIOD, -=PS_=-M1_PHASE_CONTROL and -=PS_=-M1_POLARITY. 10= line clock 11= frame clock 0 = PCLK 1 = M0SYNC 2 = LCLK 3 = FCLK

23.8.41 DC_DISP_DI_CONTROL_0

DI Control

Display Data Inversion (DI) signal generation.

This signal is typically needed to control data inversion for PWM panels and is typically output on LCD_D22 pin.

Horizontal position of this signal with respect to horizontal reference point can be programmed. DI signal together with M0 may also be used to control the actual pixel data inversion.

Pixel data may be controlled by either DI only or by (DI ^ M0) as specified by -=PS_=-PIXDATA_INV_SELECT. The inversion control signal is then used to control pixel data inversion as specified by -=PS_=-PIXDATA_INV_CONTROL. Note that even if the DI signal is disabled, pixel data inversion could still occur depending on the setting of -=PS_=-PIXDATA_INV_CONTROL.

Data inversion is limited to only active area. For the purpose of pixel data inversion, DI and M0 signals are used before the corresponding horizontal positioning so that these signals are always stable during active area.

In case M0 signal is used to control data inversion then it should be generated using line clock. M0 polarity control is not accounted when M0 is used to generate DI signal or to control pixel data inversion.

This register specifies options for DI signal as well as pixel data inversion.

Offset: 428h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
28:16	X	DI_H_POSITION: DI signal Horizontal Position. This parameter specifies the position where DI signal can toggle with respect to H reference point. It should not be programmed larger than -=PS_=-PP_H_POSITION if DI is used to control PP signal generation.

Bit	Reset	Description
7:6	X	PIXDATA_INV_CONTROL: Pixel Data Inversion Control. The control signal for pixel data inversion is defined by -=PS_=-PIXDATA_INV_SELECT 00= no pixel data inversion regardless of control signal state. 01= Pixels 0, 2, 4 ... are inverted if control signal is high. Pixels 1, 3, 5 ... are inverted if control signal is low. 10= Pixels 1, 3, 5 ... are inverted if control signal is high. Pixels 0, 2, 4 ... are inverted if control signal is low. 11= all pixel data is inverted if control signal is high. NOTE: Pixel data inversion is NOT supported for 2-pixel/3-clock 12-bit parallel display data format! 0 = NOINV 1 = EVENINV 2 = ODDINV 3 = ALLINV
4	X	PIXDATA_INV_SELECT: Pixel Data Inversion Select 0= DI signal controls pixel data inversion 1= (DI xor M0) controls pixel data inversion. 0 = DI 1 = DIXORM0
1:0	X	DI_MODE: DI signal Mode 00= DI is always low 01= DI is always high 10= DI is forced high every time M0 (before polarity adjustment) toggles from low to high; otherwise then DI toggles every line 11= DI has same frequency (phase) as M0 (before M0 polarity adjustment)

23.8.42 DC_DISP_PP_CONTROL_0

PP Control

Display Programmable Pulse (PP) signal generation

PP signal generation logic can generate up to 128 pulses per line internally and the PP pulse select registers determines which of the 128 pulses will be output. Any of the 128 internally generated pulse can be independently selected as output if they occur within one line time.

PP signal is typically output on LCD_D23 pin. Note that DI signal may impact PP generation as controlled by -=PS_=-PP_REVERSAL_CONTROL.

PP signal generation may be delayed (positioned) from H reference point (line start) controlled by -=PS_=-PP_H_DELAY. Delaying PP may cause the last few internal PP pulses to overflow to the next line.

PP is always generated using the display clock after the shift clock divider.

This register specifies options for PP signal.

Offset: 429h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:12	X	PP_LOW_PULSE: PP Low Pulse width (1 to 16)
11:8	X	PP_HIGH_PULSE: PP High Pulse width (1 to 16)
7:4	X	PP_H_DELAY: PP signal Horizontal Delay (0 to 15). This parameter specifies the position where PP signal generation starts with respect to H reference point. If DI is used to generate PP signal then this parameter should not be smaller than -=PS_=-DI_H_POSITION.

Bit	Reset	Description
3:2	X	PP_V_QUALIFIER: PP Vertical Qualifier 0= free running (not qualified) 1= V Pulse 1 qualified 2= V Pulse 2 qualified 3= V Pulse 3 qualified 0 = FREE_RUN 1 = VPULSE1 2 = VPULSE2 3 = VPULSE3
1:0	X	PP_DIRECTION: PP Direction (incrementing or decrementing) 0= always from pulse 0 to 127 (regardless of DI signal) 1= 0 to 127 if DI=0 and 127 to 0 if DI=1 2= 127 to 0 if DI=0 and 0 to 127 if DI=1 3= always 127 to 0 regardless of DI 0 = ALWAYS_INC 1 = INC_IF_DI0 2 = DEC_IF_DI0 3 = ALWAYS_DEC

23.8.43 DC_DISP_PP_SELECT_A_0

PP Select A

The next 4 registers and -=PS_=-PP_DIRECTION which of the internal 128 pulses to be output.

Each bit in the four registers corresponds to one internal pulse.

Offset: 42ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	PP_SELECT_A: PP Select bits 31 to 0

23.8.44 DC_DISP_PP_SELECT_B_0

PP Select B

Offset: 42bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	PP_SELECT_B: PP Select bits 63 to 32

23.8.45 DC_DISP_PP_SELECT_C_0

PP Select C

Offset: 42ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	PP_SELECT_C: PP Select bits 95 to 64

23.8.46 DC_DISP_PP_SELECT_D_0

PP Select D

Offset: 42dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	PP_SELECT_D: PP Select bits 127 to 96

23.8.47 DC_DISP_DISP_CLOCK_CONTROL_0

Display Clock Control

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: 42eh | Read/Write: R/W | Reset: 0b000000000110

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 de-assertion 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).

23.8.48 DC_DISP_DISP_INTERFACE_CONTROL_0

Display Interface Control

This register specifies display interface options

Offset: 42fh | Read/Write: R/W | Reset: 0b00xxxx0000

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] is blue data bits 7-2, LD[11:6] is green data bits 7-2, LD[17:12] is red data bits 7-2, LD[19:18] is blue data bits 1-0, LD[21:20] is green data bits 1-0, LD[23:22] is 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] is blue data bits 7-0, LD[15:8] is green data bits 7-0, LD[23:16] is 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 : Not supported 2 = DF1P2C18B : Not supported 3 = DF1P2C16B : Not supported 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 : Not supported 9 = DF2P1C18B : 9= 2-pixel/1-clock 18-bit parallel 10 = DFDUAL1P1C18B: Not supported

23.8.49 DC_DISP_DISP_COLOR_CONTROL_0

Display Color Control

Offset: 430h | Read/Write: R/W | Reset: 0b0000xxxxx0x0xx00xxxxxxxx0000

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
25	0x0	LCD_MD1: LCD Mode 1 signal 0 = LOW 1 = HIGH
24	0x0	LCD_MD0: LCD Mode 0 signal 0 = LOW 1 = HIGH

Bit	Reset	Description
18	0x0	NON_BASE_COLOR: Non Base Color. 0= zeros 1= ones 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 the start of frame. Typically these can be programmed in shadow register which takes effect on the next frame.
17	X	BLANK_COLOR: Blank Color. 0= zeros 1= ones Non Base Color applies to the least significant color bits which are not part of the 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
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

23.8.50 DC_DISP_SHIFT_CLOCK_OPTIONS_0

Shift Clock options

This register specifies options for both display shift clock 0 (SC0) and display shift clock 1 (SC1). SC0 signal is typically output on LCD_PCLK pin and SC1 signal is typically output on LCD_WR_N pin.

Offset: 431h | Read/Write: R/W | Reset: 0b00000000xxxxxxxx00000000

Bit	Reset	Description
23:22	0x0	SC1_CLK_DIVIDER: SC1 Clock Divider 0= divide by 1 - this is valid for all display interfaces 1= divide by 2 - this is valid only for 1-pixel/1-clock parallel display and 2-pixel/1-clock parallel display 2= divide by 4 - this is valid only for 1-pixel/1-clock parallel display 3= divide by 2, aligned with data (2p1c mode only) 0 = DIV1 1 = DIV2 2 = DIV4 3 = DIV2_ALIGNED
21:19	0x0	SC1_V_QUALIFIER: SC1 Vertical Qualifier 0= no vertical qualifier 2= vertical display active 3= 1-line extended vertical display active 4= V Pulse 1 (VP1) 5= 1-line extended V Pulse 1 others= reserved If SC1 is divided by 2 then it is synchronously reset at the beginning of the horizontal qualifier such that rising edge of SC1 is generated for the first horizontally qualified 'pixel'. If SC1 is divided by 4 then it is synchronously reset at the beginning of the horizontal qualifier such that rising edge of LCD_WR_N is generated for the second horizontally qualified 'pixel'. In the case where there is no horizontal qualifier start of horizontal display active will be used to generate the synchronous reset. If Initialization Sequence (IS) is enabled on parallel interface then only divide by 1 is allowed for SC1 Clock Divider and SC1 must have vertical and horizontal qualifiers enabled. 0 = NO_VQUAL 1 = RESERVED 2 = VACTIVE 3 = EXT_VACTIVE 4 = VPULSE1 5 = EXT_VPULSE1
18:16	0x0	SC1_H_QUALIFIER: SC1 Horizontal Qualifier 0= disable (regardless of vertical qualifier) 1= no horizontal qualifier (V qualifier only) 2= horizontal display active 3= 1-clock early & extended H display active 4= H Pulse 1 (HP1) 5= 1-clock early & extended H Pulse 1 others= reserved 0 = DISABLE 1 = NO_HQUAL 2 = HACTIVE 3 = EXT_HACTIVE 4 = HPULSE1 5 = EXT_HPULSE1

Bit	Reset	Description
7:6	0x0	SC0_CLK_DIVIDER: SC0 Clock Divider 0= divide by 1 - this is valid for all display interface 1= divide by 2 - this is valid only for 1-pixel/1-clock parallel display and 2-pixel/1-clock parallel display 2= divide by 4 - this is valid only for 1-pixel/1-clock parallel display 3= divide by 2, aligned with data (2p1c mode only) 0 = DIV1 1 = DIV2 2 = DIV4 3 = DIV2_ALIGNED
5:3	0x0	SC0_V_QUALIFIER: SC0 Vertical Qualifier 0= no vertical qualifier 2= vertical display active 3= 1-line extended vertical display active 4= V Pulse 0 (VP0) 5= 1-line extended V Pulse 0 others= reserved If SC0 is divided by 2 or 4 then it is synchronously reset at the beginning of the horizontal qualifier such that rising edge of SC0 is generated for the first horizontally qualified 'pixel'. In the case where there is no horizontal qualifier start of horizontal display active will be used to generate the synchronous reset. If Initialization Sequence (IS) is enabled on parallel interface then only divide by 1 is allowed for SC0 Clock Divider and SC0 must have vertical and horizontal qualifiers enabled. 0 = NO_VQUAL 1 = RESERVED 2 = VACTIVE 3 = EXT_VACTIVE 4 = VPULSE0 5 = EXT_VPULSE0
2:0	0x0	SC0_H_QUALIFIER: SC0 Horizontal Qualifier 0= disable (regardless of vertical qualifier) 1= no horizontal qualifier (V qualifier only) 2= horizontal display active 3= 1-clock early & extended H display active 4= H Pulse 0 (HP0) 5= 1-clock early & extended H Pulse 0 others= reserved 0 = DISABLE 1 = NO_HQUAL 2 = HACTIVE 3 = EXT_HACTIVE 4 = HPULSE0 5 = EXT_HPULSE0

23.8.51 DC_DISP_DATA_ENABLE_OPTIONS_0

Data Enable options

DE signal is display Data Enable signal which can be used to indicate valid data area and it can be output on LCD_WR_N pin if needed.

This signal can also be used to generate PCS (Parallel mode panel Chip Selecter)

- 0 = Deasserted during both VBlank and HBlank, except for the IS sequence and the gap between IS and the first active line if IS is enabled.
- 1 = Deasserted during VBlank, except for the IS line (whole IS line) if IS is enabled.
- 2 = Always asserted during refresh

Option 0 can be achieved by: DE_SELECT=ACTIVE_IS; DE_CONTROL=NORMAL

Option 1 can be achieved by: DE_SELECT=ACTIVE_IS; DE_CONTROL=ACTIVE_BLANK

Option 2 can be achieved by: DE_SELECT=ACTIVE_BLANK; DE_CONTROL=ACTIVE_BLANK

Offset: 432h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4:2	0x0	DE_CONTROL: DE (Data Enable) horizontal coverage control 0= 1-pixel clock pulse preceding active line (1-clock DE) 1= LDE active for horizontal display active time (normal DE) 2= LDE starts 1-pixel clock preceding active line but stays high on horizontal display active (early and extended DE) 3= 1-pixel clock early horizontal display active (early DE) 4= DE is active for the whole line, covering both active data and h blank (active and blank) 0 = ONECLK 1 = NORMAL 2 = EARLY_EXT 3 = EARLY 4 = ACTIVE_BLANK
1:0	0x0	DE_SELECT: DE (Data Enable) vertical coverage control 0= DE is generated on every lines (active & blank) 1= DE is generated only for active lines 2= DE is generated for active lines and Initialization sequence (if IS is enabled). DE is also asserted in the time gap between the IS and the first active line. This bit also controls STH for serial display interface in the same manner. 0 = ACTIVE_BLANK 1 = ACTIVE 2 = ACTIVE_IS

23.8.52 DC_DISP_SERIAL_INTERFACE_OPTIONS_0

Serial Display Interface Options

Controls signals for the low-voltage differential serial display interface consists of: SDT, STP and STH signals. SDT and STP are asserted high if current pixel is same as previous pixel; in this case,

SDT is toggled low sometime later but STP is either toggled low at same time as SDT (if next pixel is different than current pixel) or remains high if next pixel is same as current pixel.

When doing pixel comparison, output of dither is used, so pixel comparison depends on the base color (which maybe different than the number of output data bits).

Both SDT and STP are always low (disabled) if the pixel clock divider is 4 or less.

STH is used to indicate the beginning of line and it is asserted high once at the beginning of each line. The STH pulse exact timing width is dependent on the exact mode. STH is generated from Data Enable therefore Data Enable Select bit also controls STH generation and can be used to generate STH either only for active lines or both for active and blank lines. If STH is sent during blank lines then the blank lines are also transmitted.

Offset: 433h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7	0x0	STP_CONTROL: STP signal control 0= STP is not ORed with H Pulse 2 and vertical blank 1= STP is ORed with H Pulse 2 and vertical blank This may be set to 1 when STP needs to be forced high during blank time in which case H Pulse 2 should be programmed when STP needs to be forced high. Vertical blank is the area outside vertical display active. 0 = NORMAL 1 = EXTENDED

Bit	Reset	Description
6	0x0	STH_DURATION: STH signal duration 0= STH is high for 1 pixel clock in all cases except for 3-bit 2-channel and 6-bit 3-channel where STH is 1.5 pixel clock and for 3-bit 3-channel STH is 3 pixel clocks. 1= STH is high for 2 pixel clock in all cases except for 3-bit 3-channel STH is 4 pixel clocks. 0 = ONE_CLOCK 1 = TWO_CLOCK
5:2	0x0	SDT_STP_DURATION: SDT and STP signal duration 0= 1 shift clock 1= 1 pixel clock 2= 1 pixel clock - 1 shift clock 3= 1 pixel clock - 2 shift clock 4= 1 pixel clock - 3 shift clock 5= 1 pixel clock - 4 shift clock : : : F= 1 pixel clock - 14 shift clock STP active duration is same as SDT if next pixel is not the same as current pixel; else, STP active duration is always 1 pixel clock. Maximum valid setting is pixel clock divider - 1 for pixel clock divider > 4. If pixel clock divider is 4 or less, SDT and STP is always low.
1:0	0x0	SDT_STP_MODE: SDT and STP modes 0= SDT and STP disabled 1= reserved 2= SDT & STP enabled, duplicate data sent 3= SDT & STP enabled, duplicate data not sent 0 = DISABLE 1 = RESERVED 2 = ENABLE_DUP 3 = ENABLE

23.8.53 DC_DISP_LCD_SPI_OPTIONS_0

LCD SPI Interface Options

LCD SPI interface signals consists of:

1. SPI Clock (SCK) which can be output on LCD_SCK pin.

2. SPI Data (SDA) which can be output on LCD_SDOUT pin.

3. Optional SPI Data/Command (SDC) which can be output on LCD_DC0 pin.

4. Main-Display SPI Chip Select (Main SCS_) signal which can be output on LCS_ pin.

5. Sub-Display SPI Chip Select (Sub SCS_) signal which can be optionally output on several pins (see pin output select) - this is optional and it is used only if there is a sub display.

For LCD SPI, pixel data can only be sent to either Main-Display or Sub-Display but not to both.

Main SCS_ or Sub SCS_ signal is always active low and is typically controlled by SPI logic but can also be forced active one line prior to display active (for SIS SPI) and during vertical display active area (for LCD SPI).

Offset: 434h | Read/Write: R/W | Reset: 0b00000

Bit	Reset	Description
4	0x0	LCD_SPI_DIRECTION: LCD SPI Data Direction. Note that data direction does not affect the start byte direction (which is always msb to lsb) and position (always first 8-bit of serial data) for SPI16SB mode. 0 = MSB2LSB 1 = LSB2MSB

Bit	Reset	Description
3:2	0x0	SPI_CS_CONTROL: LCD SPI Chip Select (SCS_) Control for both IS SPI or LCD SPI 0= Main SCS_ or Sub SCS_ is controlled by LCD SPI or by IS SPI 1= Main SCS_ or Sub SCS_ is controlled by LCD SPI, and depending on LCD SPI Chip Select bit, one of them is forced active for 1-line prior to display active when IS SPI is enabled 2= Main SCS_ or Sub SCS_ is controlled by IS SPI, and depending on LCD SPI Chip Select bit, one of them is forced active during vertical display active area when LCD SPI is enabled 3= Main SCS_ or Sub SCS_, depending on LCD SPI Chip Select bit, is forced active 1-line prior to display active when IS SPI is enabled and also during vertical display active area when LCD SPI is enabled 0 = LCD_IS_SPI 1 = LCD_SPI 2 = IS_SPI 3 = FORCED
1	0x0	LCD_SPI_DC: LCD SPI Data/Command (SDC) 0= SPI Data/Command is low for LCD SPI writes to the display. For PCD6 data format, command byte is sent. 1= SPI Data/Command is high for LCD SPI writes to the display. For PCD6 data format, data byte is sent. 0 = LOW 1 = HIGH
0	0x0	LCD_SPI_CS: LCD SPI Chip Select (SCS_) 0= Send LCD SPI data to Main Display (Main SCS_ is activated) 1= Send LCD SPI data to Sub Display (Sub SCS_ is activated) This bit is also used when SPI Chip Select Control are NOT LCD_IS_SPI to determine either Main SCS_ or Sub SCS_ to be forced active. 0 = MAIN 1 = SUB

23.8.54 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: 435h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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

23.8.55 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 color key enabled. Because there are two sets of color keys, 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: 436h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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
7:0	X	COLOR_KEY0_L_R: Color Key 0 Red (V) Lower value

23.8.56 DC_DISP_COLOR_KEY0_UPPER_0

Color Key 0 Upper value

Offset: 437h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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

23.8.57 DC_DISP_COLOR_KEY1_LOWER_0

Color Key 1 Lower value

Offset: 438h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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

23.8.58 DC_DISP_COLOR_KEY1_UPPER_0

Color Key 1 Upper value

Offset: 439h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	COLOR_KEY1_U_B: Color Key 1 Blue (U) Upper value

Bit	Reset	Description
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

23.8.59 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: 43ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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

23.8.60 DC_DISP_CURSOR_BACKGROUND_0

Cursor Background color

Offset: 43dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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

23.8.61 DC_DISP_CURSOR_START_ADDR_0

Cursor Start Address

Offset: 43eh | Read/Write: R/W | Reset: 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
24	X	CURSOR_SIZE: Cursor Size 0= 32x32 1= 64x64 0 = C32X32 1 = C64X64
21:0	X	CURSOR_START_ADDR: Cursor Start Address bits 25:10

23.8.62 DC_DISP_CURSOR_START_ADDR_NS_0

Shadow of Cursor Start Address

Offset: 43fh | Read/Write: R/W | Reset: 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
24	X	CURSOR_SIZE_NS: Cursor Size 0= 32x32 1= 64x64 0 = C32X32 1 = C64X64
21:0	X	CURSOR_START_ADDR_NS: Cursor Start Address bits 25:10

23.8.63 DC_DISP_CURSOR_POSITION_0

Cursor Position

Cursor position is with respect to top-left corner of display active area, or window A, or window B, or window C as specified cursor clipping parameter.

Offset: 440h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
29:16	X	V_CURSOR_POSITION: V cursor position (signed)
13:0	X	H_CURSOR_POSITION: H cursor position (signed)

23.8.64 DC_DISP_CURSOR_POSITION_NS_0

Shadow of Cursor Position

Offset: 441h | Read/Write: R/W | Reset: 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)

23.8.65 DC_DISP_INIT_SEQ_CONTROL_0

Initialization Sequence Control

Class: Initialization Sequence (IS)

Display initialization sequence may have to be written to the display if the display has built-in frame buffer. This initialization sequence is used typically to reinitialize the display buffer start address and maybe needed once per frame (frame initialization sequence) and/or once per line (line initialization sequence).

Frame initialization sequence is sent during the horizontal active time of the line just before the first active display line. Line initialization sequence is currently NOT supported.

Initialization sequence can be done through parallel LCD interface or through SPI serial interface. Software is responsible in making sure that the active line time is sufficient to send initialization sequence.

For parallel interface initialization, the signals used as chip selects (typically these are one of the vertical signals) must be programmed to be active one line just before the first active display line.

Also SC0/SC1 clock divider must be programmed to divide by 1 if initialization sequence is enabled.

Offset: 442h | Read/Write: R/W | Reset: 0bxxxxxxxxxx00

Bit	Reset	Description
11:8	X	FRAME_INIT_SEQ_CYCLES: Frame Initialization Sequence Cycles. This parameter specifies the number of frame initialization sequence cycles to send. If programmed to 0, there is no frame initialization cycle generated.
7	X	INIT_SEQ_DC_CONTROL: Initialization Sequence DC Pin. This bit is used only for the parallel initialization sequence, and it controls how data/command is added to the vertical signal selected by -=PS_=-INIT_SEQ_DC_SIGNAL 0= parallel IS DC is inverted and then AND-ed to the vertical signal 1= parallel IS DC is ORed to the vertical signal
6:4	X	INIT_SEQ_DC_SIGNAL: Frame Initialization Sequence DC Pin. This parameter is used only for parallel initialization sequence and it specifies which signal carries the data/command signal. 0= parallel IS DC signal is not needed 1= parallel IS DC on Vertical Sync 2= parallel IS DC on Vertical Pulse 0 3= parallel IS DC on Vertical Pulse 1 4= parallel IS DC on Vertical Pulse 2 5= parallel IS DC on Vertical Pulse 3 other= reserved 0 = NODC 1 = VSYNC 2 = VPULSE0 3 = VPULSE1 4 = VPULSE2 5 = VPULSE3

Bit	Reset	Description
1	0x0	INIT_SEQUENCE_MODE: Initialization Sequence Mode 0= Send init sequence through parallel LCD interface 1= Send init sequence through SPI serial interface 0 = PLCD_INIT 1 = SPI_INIT
0	0x0	SEND_INIT_SEQUENCE: Send Initialization Sequence (IS) 0 = DISABLE 1 = ENABLE

23.8.66 DC_DISP_SPI_INIT_SEQ_DATA_A_0

SPI Init Sequence Write Data A

For parallel initialization sequence there are two possible data widths: 9 bits or 18 bits.

If parallel IS is selected, the number of bits per cycle depend on the DISP_DATA_FORMAT register programming. 18-bit parallel IS cycles are performed for 1-pixel/1-clock parallel interface (DF1P1C). 9-bit parallel IS cycles are performed for non 1-pixel/1-clock parallel interface.

Parallel IS cycles must be completed prior to the end of horizontal active of the line where IS cycles are sent. If all the cycles have been completed prior to the end of horizontal active, control signals are held inactive and last output data is held till end of horizontal active. For 9-bit parallel initialization sequence, the data is output in either LCD_D[8:0] pins or LCD_D[17:9] pins depending on display data alignment.

For serial initialization sequence using SPI (IS SPI) there are six possible data widths: 8 bits, 9 bits, 12 bits, 16 bits, 16 bits data plus start byte (24 bits), 18 bits, or 24 bits. Parameters in SPI_CONTROL register and SPI_START_BYTE register is also used for serial initialization sequence using SPI.

Serial IS cycles must also be completed prior to the end of horizontal active of the line where initialization cycles are sent. The programmer needs to make sure that register programming is such that this is true. If all the cycles have been completed prior to the end of horizontal active, SPI signals will be forced inactive until the next SPI cycles.

The following shows how initialization sequence data bits are used when sending initialization sequence:

For 9-bit parallel initialization - up to 10 initialization cycles can be done:

Init cycle	1	2	3	4	5	6	7	8	9	10
Data	8-0	17-9	26-18	35-27	44-36	53-45	62-54	71-63	80-72	89-81
LCD_PCLK enable	90	93	96	99	102	105	108	111	114	117
LCD_WR_N enable	91	94	97	100	103	106	109	112	115	118
data/command	92	95	98	101	104	107	110	113	116	119

For 18-bit parallel initialization - up to 6 initialization cycles can be done:

Init cycle	1	2	3	4	5	6
Data	17-0	35-18	53-36	71-54	89-72	107-90
LCD_PCLK enable	108	111	114	117	120	123
LCD_WR_N enable	109	112	115	118	121	124
data/command	110	113	116	119	122	125

For serial initialization using SPI, main display SPI chip select (Main SCS_) is always output on LCS_ pin while sub display SPI chip select (Sub SCS_) can be optionally output on several pins (see pin output select definition).

Initialization cycle through SPI interface can only be sent to either main or sub display but not to both and the selection bits are specified in the tables below.

Note that 0 indicates main display initialization and 1 indicates sub display initialization.

For 8-bit SPI initialization - up to 12 initialization cycles can be done:

Init cycle	1	2	3	4	5	6	7	8	9	10	11	12
Data	7-0	15-8	23-16	31-24	39-32	47-40	55-48	63-56	71-64	79-72	87-80	95-88
Main/Sub SCS_	96	98	100	102	104	106	108	110	112	114	116	118
SDC	97	99	101	103	105	107	109	111	113	115	117	119

For 12-bit SPI initialization - up to 9 initialization cycles can be done:

Init cycle	1	2	3	4	5	6	7	8	9
Data	11-0	23-12	35-24	47-36	59-48	71-60	83-72	95-84	107-96
Main/Sub SCS_	109	111	113	115	117	119	121	123	125
LCD_DC0	110	112	114	116	118	120	122	124	126

For 16-bit SPI initialization - up to 7 initialization cycles can be done:

Init cycle	1	2	3	4	5	6	7
Data	15-0	31-16	47-32	63-48	79-64	95-80	111-96
Main/Sub SCS_	112	114	116	118	120	122	124
SDC	113	115	117	119	121	123	125

For 18-bit SPI initialization - up to 6 initialization cycles can be done:

Init cycle	1	2	3	4	5	6
Data	17-0	35-18	53-36	71-54	89-72	107-90
Main/Sub SCS_	108	110	112	114	116	118
SDC	109	111	113	115	117	119

For 24-bit SPI initialization - up to 4 initialization cycles can be done:

Init cycle	1	2	3	4
Data	23-0	47-24	71-48	95-72
Main/Sub SCS_	96	98	100	102
SDC	97	99	101	103

Offset: 443h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SPI_INIT_SEQ_DATA_A: SPI Init Sequence Write Data bits 31-0

23.8.67 DC_DISP_SPI_INIT_SEQ_DATA_B_0

SPI Init Sequence Write Data B

Offset: 444h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SPI_INIT_SEQ_DATA_B: SPI Init Sequence Write Data bits 63-32

23.8.68 DC_DISP_SPI_INIT_SEQ_DATA_C_0

SPI Init Sequence Write Data C

Offset: 445h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SPI_INIT_SEQ_DATA_C: SPI Init Sequence Write Data bits 95-64

23.8.69 DC_DISP_SPI_INIT_SEQ_DATA_D_0

SPI Init Sequence Write Data D

Offset: 446h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	SPI_INIT_SEQ_DATA_D: SPI Init Sequence Write Data bits 127-96

23.8.70 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 retained for software compatibility. The clockenable fields of this register control the 2nd 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: 480h | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxx0000

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

23.8.71 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: 481h | Read/Write: R/W | Reset: 0b11001111001110000001000001011000

Bit	Reset	Description
31	ENABLE	CBR_DISPLAY0A2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBR_DISPLAY0A2MC_HYST_REQ_TH
27:24	0xf	CBR_DISPLAY0A2MC_HYST_TM
23:16	0x38	CBR_DISPLAY0A2MC_DHYST_TH
15:8	0x10	CBR_DISPLAY0A2MC_DHYST_TM
7:0	0x58	CBR_DISPLAY0A2MC_HYST_REQ_TM

23.8.72 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: 482h | Read/Write: R/W | Reset: 0b11001111000111000000100000101100

Bit	Reset	Description
31	ENABLE	CBR_DISPLAY0B2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBR_DISPLAY0B2MC_HYST_REQ_TH
27:24	0xf	CBR_DISPLAY0B2MC_HYST_TM
23:16	0x1c	CBR_DISPLAY0B2MC_DHYST_TH
15:8	0x8	CBR_DISPLAY0B2MC_DHYST_TM
7:0	0x2c	CBR_DISPLAY0B2MC_HYST_REQ_TM

23.8.73 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: 483h | Read/Write: R/W | Reset: 0b11001111001110000001000001011000

Bit	Reset	Description
31	ENABLE	CBR_DISPLAY0C2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBR_DISPLAY0C2MC_HYST_REQ_TH
27:24	0xf	CBR_DISPLAY0C2MC_HYST_TM
23:16	0x38	CBR_DISPLAY0C2MC_DHYST_TH
15:8	0x10	CBR_DISPLAY0C2MC_DHYST_TM
7:0	0x58	CBR_DISPLAY0C2MC_HYST_REQ_TM

23.8.74 DC_DISP_MCCIF_DISPLAY1B_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: 484h | Read/Write: R/W | Reset: 0b11001111000111000000100000101100

Bit	Reset	Description
31	ENABLE	CBR_DISPLAY1B2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBR_DISPLAY1B2MC_HYST_REQ_TH
27:24	0xf	CBR_DISPLAY1B2MC_HYST_TM
23:16	0x1c	CBR_DISPLAY1B2MC_DHYST_TH
15:8	0x8	CBR_DISPLAY1B2MC_DHYST_TM
7:0	0x2c	CBR_DISPLAY1B2MC_HYST_REQ_TM

23.8.75 DC_DISP_DAC_CRT_CTRL_0

CRT Control Register

Control Registers for CRT Mode. Control registers for triple DAC/CRT operation (display2tvdac signals)

A register outside of display, TVDACCCONFIG, controls which source among display/displayb/tvo goes to TVDAC.

Offset: 4c0h | Read/Write: R/W | Reset: 0b0x0x0

Bit	Reset	Description
4	0x0	NOTBLANK_SELECT: Selects the source for display2tvdac_notblank 0: notblank = d_active[10] (i.e. data_enable) 1: notblank = (lvp[1] & lhp[1]) 0 = DE 1 = LVP1_LHP1
2	0x0	SYNC_SELECT: Selects the source for display2tvdac_[hv]sync 0 = VSYNC_HSYNC 1 = LVP0_LHP0
0	0x0	OVERRIDE_NOTBLANK: If enabled, output display2tvdac_NOTBLANK is tied to 1 0 = DISABLE 1 = ENABLE

23.8.76 DC_DISP_DISP_MISC_CONTROL_0

Miscellaneous controls.

Offset: 4c1h | Read/Write: R/W | Reset: 0b10

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

23.9 Window A (WINC_A) Registers

These registers control window A parameters.

> **Note:** There are three copies of these registers for window A, B, and C. Window 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.

23.9.1 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 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.

Window A Color Palette

Offset: 500h..5ffh | Read/Write: WO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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

23.9.2 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. Supported for window A only.

Window A Palette Color Extension

Offset: 600h | Read/Write: R/W | Reset: 0bxxxxxxx

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. Bits 7-4 are used for 4-bpp mode.

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 value ranging from -4 to 3.
- Coefficients 1 and 4 are 5-bit signed value ranging from -16 to 15.
- Coefficients 2 and 3 are 8-bit unsigned value ranging from 0 to 128.
- Coefficient 0 is the multiplier for the earliest pixel (P0) in the group of 6-pixel 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.

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, 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.

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 value ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2-pixel 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 reg bits. Note that color value ranges from 0 to 255 for Y, R, G, B and -128 to 127 for U and V.

The registers under DC_WIN are double buffered.

23.9.3 DC_WIN_A_WIN_OPTIONS_0

Window A Options

Class: Display Window Settings

Display Window A parameters

Offset: 700h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxx0xxxxxxxxxxxxxxx

Bit	Reset	Description
30	0x0	A_WIN_ENABLE: Window A Window enable 0 = DISABLE 1 = ENABLE
20	X	A_DV_ENABLE: Window A Digital Vibrance Enable 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
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
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

23.9.4 DC_WIN_A_BYTE_SWAP_0

Window A Byte Swap

Offset: 701h | Read/Write: R/W | Reset: 0bxx

Bit	Reset	Description
1:0	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) 01= byte swap for each 2-byte word (2 3 0 1) 10= byte swap for each 4-byte word (0 1 2 3) 11= word swap for each 4-byte word (1 0 3 2) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW

23.9.5 DC_WIN_A_BUFFER_CONTROL_0

Window A Buffer Control

Offset: 702h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2:0	0x0	A_BUFFER_CONTROL: Window A Buffer Control 0= Host (software) controlled 1= Video Input controlled 2= Encoder Pre-Processor controlled 3= MPEG Encoder controlled 4= StretchBLT or 2D other= reserved If window buffer selection is not controlled by the host (software), then buffer start indexes are sent by the respective module specified by this parameter, and in this case, the buffer start address registers are used to specify frame stride and buffer offset for the calculated start address. 0 = HOST 1 = VI 4 = SB2D 3 = MPEGE

23.9.6 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 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.

Offset: 703h | Read/Write: R/W | Reset: 0bxxxxx

Bit	Reset	Description
4:0	X	A_COLOR_DEPTH: Window A Color Depth Supported color depths are: P1 = 1-bpp (palletized), P2 = 2-bpp (palletized), P4 = 4-bpp (palletized), 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 B6x2G6x2R6x2A8 = 32-bpp B6G6R6A8 R6x2G6x2B6x2A8 = 32-bpp R6G6B6A8 0 = P1 1 = P2 2 = P4 3 = P8 4 = B4G4R4A4 5 = B5G5R5A 6 = B5G6R5 7 = AB5G5R5 12 = B8G8R8A8 13 = R8G8B8A8 14 = B6x2G6x2R6x2A8 15 = R6x2G6x2B6x2A8 16 = YCbCr422 17 = YUV422 18 = YCbCr420P 19 = YUV420P 20 = YCbCr422P 21 = YUV422P 22 = YCbCr422R 23 = YUV422R 24 = YCbCr422RA 25 = YUV422RA

23.9.7 DC_WIN_A_POSITION_0

Window A Position

This register defines H position and size of Window A after scaling (if there is any)

Offset: 704h | Read/Write: R/W | Reset: 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.

23.9.8 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: 705h | Read/Write: R/W | Reset: 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.

23.9.9 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.

Design Note: 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.

This could be a risky calculation - maybe this should be made optional on whether we use internal hardware to calculate or left it to software to calculate.

Offset: 706h | Read/Write: R/W | Reset: 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.

23.9.10 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 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: 707h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	A_H_INITIAL_DDA: Window A H Initial DDA (4.12). This is typically programmed to 0.0

23.9.11 DC_WIN_A_V_INITIAL_DDA_0

Window A V Initial DDA

Offset: 708h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

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.

23.9.12 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_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 up-scaled and if the DDA increment is more than 1.0 then image will be down-scaled.

Offset: 709h | Read/Write: R/W | Reset: 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.
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.

23.9.13 DC_WIN_A_LINE_STRIDE_0

Window A Line Stride

Offset: 70ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
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.

23.9.14 DC_WIN_A_BUF_STRIDE_0

Window A Buffer stride

Offset: 70bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	A_BUF_STRIDE: Window A Buffer stride. Buffer stride is used to calculate the buffer addresses when the window is triggered by non-host modules. Refer to the comment of START_ADDR for programming guide. For YUV planar pixel format, this specifies buffer stride for the Y plane. The value is in bytes.

23.9.15 DC_WIN_A_BUFFER_ADDR_MODE_0

Memory Controller Tiling definitions

Offset: 70dh | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
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

23.9.16 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: 70eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
18:16	X	A_DV_CONTROL_B: Digital Vibrance control for B

Bit	Reset	Description
10:8	X	A_DV_CONTROL_G: Digital Vibrance control for G
2:0	X	A_DV_CONTROL_R: Digital Vibrance control for R

23.9.17 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 keys are 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 range (Color Key 0 and Color Key 1) can be defined and 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 and if they do the overlap 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 mode, 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, color key is compared prior to 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, color key is compared prior to horizontal/vertical scaling filter and prior to digital vibrance control.

Both upper and lower values are inclusive.

Offset: 70fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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

23.9.18 DC_WIN_A_BLEND_1WIN_0

Blend Control for this window area where it does not overlap with other windows.

Offset: 710h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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

23.9.19 DC_WIN_A_BLEND_2WIN_B_0

Blend Control for this window area that overlaps with window B only.

Offset: 711h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_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

23.9.20 DC_WIN_A_BLEND_2WIN_C_0

Blend Control for this window area that overlaps with window C only.

Offset: 712h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_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

23.9.21 DC_WIN_A_BLEND_3WIN_BC_0

Blend Control for this window area that overlaps with windows B and C only.

Offset: 713h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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.
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. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_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

23.9.22 DC_WIN_A_HP_FETCH_CONTROL_0

Window A High Priority Avoidance Fetch Parameters

This register gives extra information to the Memory Controller Client Interface (MCCIF) about the number of memory words that will be fetched per scan line and about the rate at which those words are consumed. This allows the MCCIF to more accurately arbitrate memory accesses to prevent the use of the High Priority signal. Use of this signal causes all other client accesses to be blocked in preference to the client asserting HP.

This is a state which is sometime necessary for Display as it is an isochronous client and MUST be serviced in a timely manner. However, use of this signal should be avoided if possible. These parameters help the MCCIF avoid the over-use of HP.

Note: HP fetch control is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 714h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	0x0	A_FETCH_INFO_ENABLE: Enables the sending of the Window A fetch information. For compatibility with earlier devices, this defaults to DISABLE. 0 = DISABLE : This bit should be enabled only for 12-bpp 1 = ENABLE
30:16	X	A_WORDS_PER_LINE: Window A memory fetch words per scan line. This value is in memory fetch words: Multiples of 16 bytes for Tegra 3 devices. It is computed as follows: A_WORDS_PER_LINE = (A_SIZE.A_H_SIZE * (bytes per pixel) + 15) >> 4 bytes per pixel is determined by the pixel format.
15:0	X	A_CYCLES_PER_WORD: Window A clock cycles per memory fetch word. The value of this field is essentially a measure of the data consumption rate for window A. It is computed as follows: A_CYCLES_PER_WORD = A_DDA_INCREMENT.A_H_DDA_INCREMENT / (bytes per pixel). Note that the format for this value is a fixed-point fractional value with 8 bits of integer precision and 8 bits of fractional precision. In other words, it is an '8.8' number. For example, if there is no scaling of the input image, the DDA increment will be 4096. With 32-bit RGBA pixels there will be 4 bytes per pixel, so CYCLES_PER_WORD will be ... 4096 / 4 = 1024, or 4.0 expressed in the 8.8 format. Any scaling performed on the pixels will change the rate at which pixels are consumed. Scaling up will increase the value of DDA increment and will therefore increase the number of cycles between memory fetches. Conversely, scaling down will decrease the value of DDA increment and memory fetches will occur more frequently.

23.9.23 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-bit per pixel alpha are quantized back to 1- and 4-bits, respectively, after scaling. For 1-bit alpha case, finer scaling can be achieved by changing BLEND_WEIGHT* instead.

Offset: 715h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxx

Bit	Reset	Description
16	DISABLE	A_GLOBAL_ALPHA_ENABLE: When enabled, color palette *must* also be enabled and programmed. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
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.

23.10 WINBUF_A Registers

The registers under DC_WINBUF are triple-buffered.

23.10.1 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 specify 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 3-ii 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 or linear mode. However, for linear mode, the addresses calculated are real physical addresses, but for tile mode, these addresses will be translated to the real physical addresses by the memory controller client before used.

- When a window is host triggered, starting address of a window is calculated as follows by HW.
  - 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 HW.
- 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 HW.

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 multiple of 2 in yuv planar format (the last bit is ignored), but with no restrictions on other color formats.

- ADDR_V_OFFSET has no restrictions

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 formula in 2-i,ii 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 multiple of 2. The least significant bit of H_OFFSTE is ignored.

- For 32-bpp formats,

-- (START_ADDR+H_OFFSTE) needs to be multiple of 4. The least two significant bits of H_OFFSTE are ignored.

- 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 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 multiple of 4.

-- ADDR_H_OFFSET: Needs to be multiple of 2. If needs to point to odd pixel position, program ADDR_H_OFFSET to be the previous position (or the next position if H-flipped) and program H_INITIAL_DDA bit 12 to 1.

-- ADDR_V_OFFSET: Needs to be multiple of 2. If needs to point to odd line number, program ADDR_V_OFFSTE to be the previous line number (or next line number if V-flipped) and program V_INITIAL_DDA bit 12 to 1.

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: 800h | Read/Write: R/W | Reset: 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.

23.10.2 DC_WINBUF_A_START_ADDR_NS_0

Window A Shadowed Start Address

Offset: 801h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	A_START_ADDR_NS: Window A Shadowed Start Address. This is ARM set shadow of Start Address.

23.10.3 DC_WINBUF_A_ADDR_H_OFFSET_0

Window A Horizontal address offset

Offset: 806h | Read/Write: R/W | Reset: 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 HW.

23.10.4 DC_WINBUF_A_ADDR_H_OFFSET_NS_0

Window A Shadowed Horizontal address offset

Offset: 807h | Read/Write: R/W | Reset: 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

23.10.5 DC_WINBUF_A_ADDR_V_OFFSET_0

Window A Vertical address offset

Offset: 808h | Read/Write: R/W | Reset: 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 LSB 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 HW.

23.10.6 DC_WINBUF_A_ADDR_V_OFFSET_NS_0

Window A Shadowed Vertical address offset

Offset: 809h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	A_ADDR_V_OFFSET_NS: Window A Shadowed Vertical address offset. This is ARM set shadow of ADDR_V_OFFSET

23.10.7 DC_WINBUF_A_UFLOW_STATUS

Window A FIFO Underflow Status Register

Offset: 80ah | Read/Write: R/W | Reset: 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.

23.11 Window B (WINC_B) Registers

These registers control window B parameters

23.11.1 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: 500h..5ffh | Read/Write: WO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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

23.11.2 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. Supported for window A only.

Offset: 600h | Read/Write: R/W | Reset: 0bxxxxxxx

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, and bits 7-4 are used for 4-bpp mode.

23.11.3 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 value ranging from -4 to 3.
- Coefficients 1 and 4 are 5-bit signed value ranging from -16 to 15.
- Coefficients 2 and 3 are 8-bit unsigned value ranging from 0 to 128.

- Coefficient 0 is the multiplier for the earliest pixel (P0) in the group of 6-pixel 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: 601h | Read/Write: R/W | Reset: 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)

23.11.4 DC_WINC_B_H_FILTER_P01_0

Window B Horizontal Filter phase 01

Offset: 602h | Read/Write: R/W | Reset: 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)

23.11.5 DC_WINC_B_H_FILTER_P02_0

Window B Horizontal Filter phase 02

Offset: 603h | Read/Write: R/W | Reset: 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)

23.11.6 DC_WINC_B_H_FILTER_P03_0

Window B Horizontal Filter phase 03

Offset: 604h | Read/Write: R/W | Reset: 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)

23.11.7 DC_WINC_B_H_FILTER_P04_0

Window B Horizontal Filter phase 04

Offset: 605h | Read/Write: R/W | Reset: 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)

23.11.8 DC_WINC_B_H_FILTER_P05_0

Window B Horizontal Filter phase 05

Offset: 606h | Read/Write: R/W | Reset: 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)

23.11.9 DC_WINC_B_H_FILTER_P06_0

Window B Horizontal Filter phase 06

Offset: 607h | Read/Write: R/W | Reset: 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)

23.11.10 DC_WINC_B_H_FILTER_P07_0

Window B Horizontal Filter phase 07

Offset: 608h | Read/Write: R/W | Reset: 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)

23.11.11 DC_WINC_B_H_FILTER_P08_0

Window B Horizontal Filter phase 08

Offset: 609h | Read/Write: R/W | Reset: 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)

23.11.12 DC_WINC_B_H_FILTER_P09_0

Window B Horizontal Filter phase 09

Offset: 60ah | Read/Write: R/W | Reset: 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)

23.11.13 DC_WINC_B_H_FILTER_P0A_0

Window B Horizontal Filter phase 0A

Offset: 60bh | Read/Write: R/W | Reset: 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)

23.11.14 DC_WINC_B_H_FILTER_P0B_0

Window B Horizontal Filter phase 0B

Offset: 60ch | Read/Write: R/W | Reset: 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)

23.11.15 DC_WINC_B_H_FILTER_P0C_0

Window B Horizontal Filter phase 0C

Offset: 60dh | Read/Write: R/W | Reset: 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)

23.11.16 DC_WINC_B_H_FILTER_P0D_0

Window B Horizontal Filter phase 0D

Offset: 60eh | Read/Write: R/W | Reset: 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)

23.11.17 DC_WINC_B_H_FILTER_P0E_0

Window B Horizontal Filter phase 0E

Offset: 60fh | Read/Write: R/W | Reset: 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)

23.11.18 DC_WINC_B_H_FILTER_P0F_0

Window B Horizontal Filter phase 0F

Offset: 610h | Read/Write: R/W | Reset: 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)

23.11.19 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: 611h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_CSC_YOF: Y Offset in s.7.0 format

23.11.20 DC_WINC_B_CSC_KYRGB_0

Window B CSC Y Coefficient (gain) for RGB

Offset: 612h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	B_CSC_KYRGB: Y Gain for R, G, B colors in 2.8 format

23.11.21 DC_WINC_B_CSC_KUR_0

Window B CSC U coefficient for R

Offset: 613h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	B_CSC_KUR: U coefficients for R in s.2.8 format

23.11.22 DC_WINC_B_CSC_KVR_0

Window B CSC V coefficient for R

Offset: 614h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	B_CSC_KVR: V coefficients for R in s.2.8 format

23.11.23 DC_WINC_B_CSC_KUG_0

Window B CSC U coefficient for G

Offset: 615h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	B_CSC_KUG: U coefficients for G in s.1.8 format

23.11.24 DC_WINC_B_CSC_KVG_0

Window B CSC V coefficient for G

Offset: 616h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	B_CSC_KVG: V coefficients for G in s.1.8 format

23.11.25 DC_WINC_B_CSC_KUB_0

Window B CSC U coefficient for B

Offset: 617h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	B_CSC_KUB: U coefficients for B in s.2.8 format

23.11.26 DC_WINC_B_CSC_KVB_0

Window B CSC V coefficient for B

Offset: 618h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	B_CSC_KVB: V coefficients for B in s.2.8 format

23.11.27 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 value ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2-pixel 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: 619h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P00C0: Phase 00 coefficient 0 (typically 128)

23.11.28 DC_WINC_B_V_FILTER_P01_0

Window B Vertical Filter phase 01

Offset: 61ah | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P01C0: Phase 01 coefficient 0 (typically 120)

23.11.29 DC_WINC_B_V_FILTER_P02_0

Window B Vertical Filter phase 02

Offset: 61bh | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P02C0: Phase 02 coefficient 0 (typically 112)

23.11.30 DC_WINC_B_V_FILTER_P03_0

Window B Vertical Filter phase 03

Offset: 61ch | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P03C0: Phase 03 coefficient 0 (typically 104)

23.11.31 DC_WINC_B_V_FILTER_P04_0

Window B Vertical Filter phase 04

Offset: 61dh | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P04C0: Phase 04 coefficient 0 (typically 96)

23.11.32 DC_WINC_B_V_FILTER_P05_0

Window B Vertical Filter phase 05

Offset: 61eh | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P05C0: Phase 05 coefficient 0 (typically 88)

23.11.33 DC_WINC_B_V_FILTER_P06_0

Window B Vertical Filter phase 06

Offset: 61fh | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P06C0: Phase 06 coefficient 0 (typically 80)

23.11.34 DC_WINC_B_V_FILTER_P07_0

Window B Vertical Filter phase 07

Offset: 620h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P07C0: Phase 07 coefficient 0 (typically 72)

23.11.35 DC_WINC_B_V_FILTER_P08_0

Window B Vertical Filter phase 08

Offset: 621h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P08C0: Phase 08 coefficient 0 (typically 64)

23.11.36 DC_WINC_B_V_FILTER_P09_0

Window B Vertical Filter phase 09

Offset: 622h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P09C0: Phase 09 coefficient 0 (typically 56)

23.11.37 DC_WINC_B_V_FILTER_P0A_0

Window B Vertical Filter phase 0A

Offset: 623h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P0AC0: Phase 0A coefficient 0 (typically 48)

23.11.38 DC_WINC_B_V_FILTER_P0B_0

Window B Vertical Filter phase 0B

Offset: 624h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P0BC0: Phase 0B coefficient 0 (typically 40)

23.11.39 DC_WINC_B_V_FILTER_P0C_0

Window B Vertical Filter phase 0C

Offset: 625h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P0CC0: Phase 0C coefficient 0 (typically 32)

23.11.40 DC_WINC_B_V_FILTER_P0D_0

Window B Vertical Filter phase 0D

Offset: 626h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P0DC0: Phase 0D coefficient 0 (typically 24)

23.11.41 DC_WINC_B_V_FILTER_P0E_0

Window B Vertical Filter phase 0E\

Offset: 627h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P0EC0: Phase 0E coefficient 0 (typically 16)

23.11.42 DC_WINC_B_V_FILTER_P0F_0

Window B Vertical Filter phase 0F

Offset: 628h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	B_V_FILTER_P0FC0: Phase 0F coefficient 0 (typically 8)

The registers under DC_WIN are double buffered

23.11.43 DC_WIN_B_WIN_OPTIONS_0

Window B Options

Class: Display Window Settings

Display Window B parameters

Offset: 700h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxx0x0xxxxxxxxxxxxxx

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 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. 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
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

23.11.44 DC_WIN_B_BYTE_SWAP_0

Window B Byte Swap

Offset: 701h | Read/Write: R/W | Reset: 0bxx

Bit	Reset	Description
1:0	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. 00= no byte swap (3 2 1 0) 01= byte swap for each 2-byte word (2 3 0 1) 10= byte swap for each 4-byte word (0 1 2 3) 11= word swap for each 4-byte word (1 0 3 2) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW

23.11.45 DC_WIN_B_BUFFER_CONTROL_0

Window B Buffer Control

Offset: 702h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2:0	0x0	B_BUFFER_CONTROL: Window B Buffer Control 0= Host (software) controlled 1= Video Input controlled 2= Encoder Pre-Processor controlled 3= MPEG Encoder controlled 4= StretchBLT or 2D other= reserved If window buffer selection is not controlled by host (software) then buffer start indexes are sent by the respective module specified by this parameter, and in this case, the buffer start address registers are used to specify frame stride and buffer offset for the calculated start address. 0 = HOST 1 = VI 2 = EPP 4 = SB2D 3 = MPEGE

23.11.46 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 same as YCbCr422R in memory and YUV422RA is 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.

Offset: 703h | Read/Write: R/W | Reset: 0bxxxxx

Bit	Reset	Description
4:0	X	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 B6x2G6x2R6x2A8 = 32-bpp B6G6R6A8 R6x2G6x2B6x2A8 = 32-bpp R6G6B6A8 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 0 = P1 1 = P2 2 = P4 3 = P8 4 = B4G4R4A4 5 = B5G5R5A 6 = B5G6R5 7 = AB5G5R5 12 = B8G8R8A8 13 = R8G8B8A8 14 = B6x2G6x2R6x2A8 15 = R6x2G6x2B6x2A8 16 = YCbCr422 17 = YUV422 18 = YCbCr420P 19 = YUV420P 20 = YCbCr422P 21 = YUV422P 22 = YCbCr422R 23 = YUV422R 24 = YCbCr422RA 25 = YUV422RA

23.11.47 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: 704h | Read/Write: R/W | Reset: 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.

23.11.48 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 which affects performance.

Offset: 705h | Read/Write: R/W | Reset: 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.

23.11.49 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.

Design Note: 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.

This could be a risky calculation - maybe this should be made optional on whether we use internal hardware to calculate or left it to software to calculate.

Offset: 706h | Read/Write: R/W | Reset: 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.

23.11.50 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. Similar with the V Initial DDA.

Offset: 707h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	B_H_INITIAL_DDA: Window B H Initial DDA (4.12). This is typically programmed to 0.0

23.11.51 DC_WIN_B_V_INITIAL_DDA_0

Window B V Initial DDA

Offset: 708h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

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.

23.11.52 DC_WIN_B_DDA_INCREMENT_0

Window B 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 2 Bytes/pix formats.

8.0 (0x8000) for 4 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 up-scaled and if DDA increment is more than 1.0 then image will be down-scaled.

Offset: 709h | Read/Write: R/W | Reset: 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.

23.11.53 DC_WIN_B_LINE_STRIDE_0

Window B Line Stride

Offset: 70ah | Read/Write: R/W | Reset: 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.

Bit	Reset	Description
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 details.

23.11.54 DC_WIN_B_BUF_STRIDE_0

Window B Buffer stride

Offset: 70bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_BUF_STRIDE: Window B Buffer stride. Buffer stride is used to calculate the buffer addresses when the window is triggered by non-host modules. Refer to the comment of of START_ADDR for programming guide. For YUV planar pixel format, this specifies buffer stride for the Y plane. The value is in bytes.

23.11.55 DC_WIN_B_UV_BUF_STRIDE_0

Window B Buffer stride for U/V plane

Offset: 70ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_UV_BUF_STRIDE: Window B. This value is in bytes. For YUV planar pixel format, this specifies buffer stride for the U/V plane.

23.11.56 DC_WIN_B_BUFFER_ADDR_MODE_0

Memory Controller Tiling definitions

Offset: 70dh | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxx0

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. 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

23.11.57 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: 70eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxx

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

23.11.58 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 window. If more than 1 windows color keys are 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 range (Color Key 0 and Color Key 1) can be defined and 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 and if they do the overlap 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 mode, 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, color key is compared prior to 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, color key is compared prior to horizontal/vertical scaling filter and prior to digital vibrance control. Both upper and lower values are inclusive.

Offset: 70fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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

23.11.59 DC_WIN_B_BLEND_1WIN_0

Blend Control for this window area where it does not overlap with other windows.

Offset: 710h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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. 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	B_CKEY_ENABLE_1WIN: Window color key enable 0 = Color Key 0 and 1 Disable 1 = Color Key 0 Enable

Bit	Reset	Description
		2 = Color Key 1 Enable 3 = Color Key 0 and 1 Enable 0 = NOKEY 1 = CKEY0 2 = CKEY1 3 = CKEY01

23.11.60 DC_WIN_B_BLEND_2WIN_A_0

Blend Control for this window area that overlaps with window A only.

Offset: 711h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_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 1 = CKEY0 2 = CKEY1 3 = CKEY01

23.11.61 DC_WIN_B_BLEND_2WIN_C_0

Blend Control for this window area that overlaps with window C only.

Offset: 712h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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.

Bit	Reset	Description
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. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_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

23.11.62 DC_WIN_B_BLEND_3WIN_AC_0

Blend Control for this window area that overlaps with windows A & C only.

Offset: 713h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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 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. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_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

23.11.63 DC_WIN_B_HP_FETCH_CONTROL_0

Window B High Priority Avoidance Fetch Parameters

This register gives extra information to the Memory Controller Client Interface (MCCIF) about the number of memory words that will be fetched per scan line and about the rate at which those words are consumed. This allows the MCCIF to more accurately arbitrate memory accesses to prevent the use of the High Priority signal. Use of this signal causes all other client accesses to be blocked in preference to the client asserting HP. Obviously, this is a state which is sometime necessary for Display since it is an isochronous client and MUST be serviced in a timely manner. However, use of this signal should be avoided if possible. These parameters help the MCCIF avoid the over-use of HP.

Note: HP fetch control is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 714h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	0x0	B_FETCH_INFO_ENABLE: Enables the sending of the Window B fetch information. For compatibility with earlier devices, this defaults to DISABLE. 0 = DISABLE : This bit should be enabled only for 12-bpp 1 = ENABLE
30:16	X	B_WORDS_PER_LINE: Window B memory fetch words per scan line. This value is in memory fetch words: Multiples of 16 bytes for Tegra 3 devices. It is computed as follows: B_WORDS_PER_LINE = (B_SIZE.B_H_SIZE * (bytes per pixel) + 15) >> 4 bytes per pixel is determined by the pixel format.
15:0	X	B_CYCLES_PER_WORD: Window B clock cycles per memory fetch word. The value of this field is essentially a measure of the data consumption rate for window B. It is computed as follows: B_CYCLES_PER_WORD = B_DDA_INCREMENT.B_H_DDA_INCREMENT / (bytes per pixel). Note that the format for this value is a fixed-point fractional value with 8 bits of integer precision and 8 bits of fractional precision. In other words, it is an '8.8' number. For example, if there is no scaling of the input image, the DDA increment will be 4096. With 32-bit RGBA pixels there will be 4 bytes per pixel, so CYCLES_PER_WORD will be ... 4096 / 4 = 1024, or 4.0 expressed in the 8.8 format. Any scaling performed on the pixels will change the rate at which pixels are consumed. Scaling up will increase the value of DDA increment and will therefore increase the number of cycles between memory fetches. Conversely, scaling down will decrease the value of DDA increment and memory fetches will occur more frequently.

23.11.64 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-bit per pixel alpha are quantized back to 1- and 4-bits, respectively, after scaling. For 1-bit alpha case, finer scaling can be achieved by changing BLEND_WEIGHT* instead.

Offset: 715h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxx

Bit	Reset	Description
16	DISABLE	B_GLOBAL_ALPHA_ENABLE: When enabled, the 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.

23.12 WINBUF_B Registers

The registers under DC_WINBUF are triple-buffered.

23.12.1 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 specify 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 3-ii 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 or linear mode. However, for linear mode, the addresses calculated are real physical addresses, but for tile mode, these addresses will be translated to the real physical addresses by the memory controller client before used.

- When a window is host triggered, starting address of a window is calculated as follows by HW.
  - 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 HW.
- 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 HW.

    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 multiple of 2 in yuv planar format (the last bit is ignored), but with no restrictions on other color formats.
  - ADDR_V_OFFSET has no restrictions

- 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 formula in 2-i,ii 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 multiple of 2. The least significant bit of H_OFFSTE is ignored.
  - For 32-bpp formats,

    (START_ADDR+H_OFFSTE) needs to be multiple of 4. The least two significant bits of H_OFFSTE are ignored.
  - 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 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 multiple of 4.

    ADDR_H_OFFSET:

    Needs to be multiple of 2. If needs to point to odd pixel position, program ADDR_H_OFFSET to be the previous position (or the next position if H-flipped) and program H_INITIAL_DDA bit 12 to 1.

    ADDR_V_OFFSET:

    Needs to be multiple of 2. If needs to point to odd line number, program ADDR_V_OFFSTE to be the previous line number (or next line number if V-flipped) and program V_INITIAL_DDA bit 12 to 1.

- 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, the frame start and frame end flag are active every time a buffer index is sent.

Offset: 800h | Read/Write: R/W | Reset: 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 LSB are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

23.12.2 DC_WINBUF_B_START_ADDR_NS_0

Window B Shadowed Start Address

Offset: 801h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_START_ADDR_NS: Window B Shadowed Start Address. This is ARM set shadow of Start Address.

23.12.3 DC_WINBUF_B_START_ADDR_U_0

Window B Start Address for U plane

Offset: 802h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_START_ADDR_U: Window B Start Address for U plane. This is a byte address.

23.12.4 DC_WINBUF_B_START_ADDR_U_NS_0

Window B Shadowed Start Address for U plane

Offset: 803h | Read/Write: R/W | Reset: 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

23.12.5 DC_WINBUF_B_START_ADDR_V_0

Window B Start Address for V plane

Offset: 804h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_START_ADDR_V: Window B Start Address for V plane. This is a byte address.

23.12.6 DC_WINBUF_B_START_ADDR_V_NS_0

Window B Shadowed Start Address for V plane

Offset: 805h | Read/Write: R/W | Reset: 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

23.12.7 DC_WINBUF_B_ADDR_H_OFFSET_0

Window B Horizontal address offset

Offset: 806h | Read/Write: R/W | Reset: 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 LSB 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 HW.

23.12.8 DC_WINBUF_B_ADDR_H_OFFSET_NS_0

Window B Shadowed Horizontal address offset

Offset: 807h | Read/Write: R/W | Reset: 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

23.12.9 DC_WINBUF_B_ADDR_V_OFFSET_0

Window B Vertical address offset

Offset: 808h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	B_ADDR_V_OFFSET: Window B Vertical address offset. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSB are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies vertical offset of Y plane. Vertical offsets of U/V plane is derived by HW.

23.12.10 DC_WINBUF_B_ADDR_V_OFFSET_NS_0

Window B Shadowed Vertical address offset

Offset: 809h | Read/Write: R/W | Reset: 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

23.12.11 DC_WINBUF_B_UFLOW_STATUS_0

Window B FIFO Underflow Status Register

Offset: 80ah | Read/Write: R/W | Reset: 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.

23.13 Window C (WIN_C) Registers

These registers control window C parameters

23.13.1 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: 500h..5ffh | Read/Write: WO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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

23.13.2 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. Supported for window A only.

Offset: 600h | Read/Write: R/W | Reset: 0bxxxxxxx

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.

23.13.3 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-pixel 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: 601h | Read/Write: R/W | Reset: 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)

23.13.4 DC_WINC_C_H_FILTER_P01_0

Window C Horizontal Filter phase 01

Offset: 602h | Read/Write: R/W | Reset: 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)

23.13.5 DC_WINC_C_H_FILTER_P02_0

Window C Horizontal Filter phase 02

Offset: 603h | Read/Write: R/W | Reset: 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)

23.13.6 DC_WINC_C_H_FILTER_P03_0

Window C Horizontal Filter phase 03

Offset: 604h | Read/Write: R/W | Reset: 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)

23.13.7 DC_WINC_C_H_FILTER_P04_0

Window C Horizontal Filter phase 04

Offset: 605h | Read/Write: R/W | Reset: 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)

23.13.8 DC_WINC_C_H_FILTER_P05_0

Window C Horizontal Filter phase 05

Offset: 606h | Read/Write: R/W | Reset: 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)

23.13.9 DC_WINC_C_H_FILTER_P06_0

Window C Horizontal Filter phase 06

Offset: 607h | Read/Write: R/W | Reset: 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)

23.13.10 DC_WINC_C_H_FILTER_P07_0

Window C Horizontal Filter phase 07

Offset: 608h | Read/Write: R/W | Reset: 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)

23.13.11 DC_WINC_C_H_FILTER_P08_0

Window C Horizontal Filter phase 08

Offset: 609h | Read/Write: R/W | Reset: 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)

23.13.12 DC_WINC_C_H_FILTER_P09_0

Window C Horizontal Filter phase 09

Offset: 60ah | Read/Write: R/W | Reset: 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)

23.13.13 DC_WINC_C_H_FILTER_P0A_0

Window C Horizontal Filter phase 0A

Offset: 60bh | Read/Write: R/W | Reset: 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)

23.13.14 DC_WINC_C_H_FILTER_P0B_0

Window C Horizontal Filter phase 0B

Offset: 60ch | Read/Write: R/W | Reset: 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)

23.13.15 DC_WINC_C_H_FILTER_P0C_0

Window C Horizontal Filter phase 0C

Offset: 60dh | Read/Write: R/W | Reset: 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)

23.13.16 DC_WINC_C_H_FILTER_P0D_0

Window C Horizontal Filter phase 0D

Offset: 60eh | Read/Write: R/W | Reset: 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)

23.13.17 DC_WINC_C_H_FILTER_P0E_0

Window C Horizontal Filter phase 0E

Offset: 60fh | Read/Write: R/W | Reset: 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)

23.13.18 DC_WINC_C_H_FILTER_P0F_0

Window C Horizontal Filter phase 0F

Offset: 610h | Read/Write: R/W | Reset: 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)

23.13.19 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, 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: 611h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	C_CSC_YOF: Y Offset in s.7.0 format

23.13.20 DC_WINC_C_CSC_KYRGB_0

Window C CSC Y Coefficient (gain) for RGB

Offset: 612h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	C_CSC_KYRGB: Y Gain for R, G, B colors in 2.8 format

23.13.21 DC_WINC_C_CSC_KUR_0

Window C CSC U coefficient for R

Offset: 613h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
10:0	X	C_CSC_KUR: U coefficients for R in s.2.8 format

23.13.22 DC_WINC_C_CSC_KVR_0

Window C CSC V coefficient for R

Offset: 614h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	C_CSC_KVR: V coefficients for R in s.2.8 format

23.13.23 DC_WINC_C_CSC_KUG_0

Window C CSC U coefficient for G

Offset: 615h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	C_CSC_KUG: U coefficients for G in s.1.8 format

23.13.24 DC_WINC_C_CSC_KVG_0

Window C CSC V coefficient for G

Offset: 616h | Read/Write: R/W | Reset: 0bxxxxxxxxxx

Bit	Reset	Description
9:0	X	C_CSC_KVG: V coefficients for G in s.1.8 format

23.13.25 DC_WINC_C_CSC_KUB_0

Window C CSC U coefficient for B

Offset: 617h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	C_CSC_KUB: U coefficients for B in s.2.8 format

23.13.26 DC_WINC_C_CSC_KVB_0

Window C CSC V coefficient for B

Offset: 618h | Read/Write: R/W | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10:0	X	C_CSC_KVB: V coefficients for B in s.2.8 format

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 value ranging from 0 to 128.

Coefficient 0 is the multiplier for the earlier pixel (P0) in the group of 2-pixel 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.

The registers under DC_WIN are double buffered.

23.13.27 DC_WIN_C_WIN_OPTIONS_0

Window C Options

Class: Display Window Settings

Display Window C parameters

Offset: 700h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxx0xxxxxxxxxxxxxxx

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
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

Bit	Reset	Description
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

23.13.28 DC_WIN_C_BYTE_SWAP_0

Window C Byte Swap

Offset: 701h | Read/Write: R/W | Reset: 0bxx

Bit	Reset	Description
1:0	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. 00= no byte swap (3 2 1 0) 01= byte swap for each 2-byte word (2 3 0 1) 10= byte swap for each 4-byte word (0 1 2 3) 11= word swap for each 4-byte word (1 0 3 2) 0 = NOSWAP 1 = SWAP2 2 = SWAP4 3 = SWAP4HW

23.13.29 DC_WIN_C_BUFFER_CONTROL_0

Window C Buffer Control

Offset: 702h | Read/Write: R/W | Reset: 0b000

Bit	Reset	Description
2:0	0x0	C_BUFFER_CONTROL: Window C Buffer Control 0= Host (software) controlled 1= Video Input controlled 2= Encoder Pre-Processor controlled 3= MPEG Encoder controlled 4= StretchBLT or 2D other= reserved If window buffer selection is not controlled by host (software) then buffer start indexes are sent by the respective module specified by this parameter, and in this case, the buffer start address registers are used to specify frame stride and buffer offset for the calculated start address. 0 = HOST 1 = VI 2 = EPP 4 = SB2D 3 = MPEGE

23.13.30 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 same as YCbCr422R in memory and YUV422RA is 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 lsb zeroed out. R6x2G6x2B6x2A8 is similar to R8G8B6A8 but with the 2 lsb zeroed out.

Offset: 703h | Read/Write: R/W | Reset: 0bxxxxx

Bit	Reset	Description
4:0	X	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 B6x2G6x2R6x2A8 = 32-bpp B6G6R6A8 R6x2G6x2B6x2A8 = 32-bpp R6G6B6A8 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 0 = P1 1 = P2 2 = P4 3 = P8 4 = B4G4R4A4 5 = B5G5R5A 6 = B5G6R5 7 = AB5G5R5 12 = B8G8R8A8 13 = R8G8B8A8 14 = B6x2G6x2R6x2A8 15 = R6x2G6x2B6x2A8 16 = YCbCr422 17 = YUV422 18 = YCbCr420P 19 = YUV420P 20 = YCbCr422P 21 = YUV422P 22 = YCbCr422R 23 = YUV422R 24 = YCbCr422RA 25 = YUV422RA

23.13.31 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: 704h | Read/Write: R/W | Reset: 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.

23.13.32 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.

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: 705h | Read/Write: R/W | Reset: 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.

23.13.33 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.

Design Note: 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.

This could be a risky calculation - maybe this should be made optional on whether we use internal hardware to calculate or left it to software to calculate.

Offset: 706h | Read/Write: R/W | Reset: 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.

23.13.34 DC_WIN_C_H_INITIAL_DDA_0

Window C 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: 707h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	C_H_INITIAL_DDA: Window C H Initial DDA (4.12) This is typically programmed to 0.0

23.13.35 DC_WIN_C_V_INITIAL_DDA_0

Window C V Initial DDA

Offset: 708h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

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.

23.13.36 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 2 Bytes/pix formats.

8.0 (0x8000) for 4 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 up-scaled and if DDA increment is more than 1.0 then image will be down-scaled.

Offset: 709h | Read/Write: R/W | Reset: 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.

23.13.37 DC_WIN_C_LINE_STRIDE_0

Window C Line Stride

Offset: 70ah | Read/Write: R/W | Reset: 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.

23.13.38 DC_WIN_C_BUF_STRIDE_0

Window C Buffer stride

Offset: 70bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_BUF_STRIDE: Window C Buffer stride. Buffer stride is used to calculate the buffer addresses when the window is triggered by non-host modules. Refer to the comment of START_ADDR for programming guide. For YUV planar pixel format, this specifies buffer stride for the Y plane. The value is in bytes.

23.13.39 DC_WIN_C_UV_BUF_STRIDE_0

Window C Buffer stride for U/V plane

Offset: 70ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_UV_BUF_STRIDE: Window C. This value is in bytes. For YUV planar pixel format, this specifies buffer stride for the U/V plane.

23.13.40 DC_WIN_C_BUFFER_ADDR_MODE_0

Memory Controller Tiling definitions

Offset: 70dh | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxx0

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

23.13.41 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: 70eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxx

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

23.13.42 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 windows color keys are 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 range (Color Key 0 and Color Key 1) can be defined and 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 and if they do the overlap 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 mode, 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, color key is compared prior to 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, color key is compared prior to horizontal/vertical scaling filter and prior to digital vibrance control.

Both upper and lower values are inclusive.

Offset: 70fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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

23.13.43 DC_WIN_C_BLEND_1WIN_0

Blend Control for this window area where it does not overlap with other windows.

Offset: 710h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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

23.13.44 DC_WIN_C_BLEND_2WIN_A_0

Blend Control for this window area that overlaps with window A only.

Offset: 711h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_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

23.13.45 DC_WIN_C_BLEND_2WIN_B_0

Blend Control for this window area that overlaps with window B only.

Offset: 712h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_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

23.13.46 DC_WIN_C_BLEND_3WIN_AB_0

Blend Control for this window area that overlaps with windows A & B only.

Offset: 713h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxx

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.
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

Bit	Reset	Description
		have this setting. 0 = FIX_WEIGHT 1 = ALPHA_WEIGHT 2 = DEPENDENT_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

23.13.47 DC_WIN_C_HP_FETCH_CONTROL_0

Window C High Priority Avoidance Fetch Parameters

This register gives extra information to the Memory Controller Client Interface (MCCIF) about the number of memory words that will be fetched per scan line and about the rate at which those words are consumed. This allows the MCCIF to more accurately arbitrate memory accesses to prevent the use of the High Priority signal.

Use of this signal causes all other client accesses to be blocked in preference to the client asserting HP. Obviously, this is a state which is sometime necessary for Display since it is an isochronous client and MUST be serviced in a timely manner. However, use of this signal should be avoided if possible. These parameters help the MCCIF avoid the over-use of HP.

Note: HP fetch control is no longer supported by the MCCIF clients. Registers are retained for software compatibility, but are not used by the hardware.

Offset: 714h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	0x0	C_FETCH_INFO_ENABLE: Enables the sending of the Window C fetch information. For compatibility with earlier devices, this defaults to DISABLE. 0 = DISABLE : This bit should be enabled only for 12-bpp 1 = ENABLE
30:16	X	C_WORDS_PER_LINE: Window C memory fetch words per scan line. This value is in memory fetch words: Multiples of 16 bytes for Tegra 3 devices. It is computed as follows: C_WORDS_PER_LINE = (C_SIZE.C_H_SIZE * (bytes per pixel) + 15) >> 4 bytes per pixel is determined by the pixel format.
15:0	X	C_CYCLES_PER_WORD: Window C clock cycles per memory fetch word. The value of this field is essentially a measure of the data consumption rate for window C. It is computed as follows: C_CYCLES_PER_WORD = C_DDA_INCREMENT.C_H_DDA_INCREMENT / (bytes per pixel). Note that the format for this value is a fixed-point fractional value with 8 bits of integer precision and 8 bits of fractional precision. In other words, it is an '8.8' number. For example, if there is no scaling of the input image, the DDA increment will be 4096. With 32-bit RGBA pixels there will be 4 bytes per pixel, so CYCLES_PER_WORD will be ... 4096 / 4 = 1024, or 4.0 expressed in the 8.8 format. Any scaling performed on the pixels will change the rate at which pixels are consumed. Scaling up will increase the value of DDA increment and will therefore increase the number of cycles between memory fetches. Conversely, scaling down will decrease the value of DDA increment and memory fetches will occur more frequently.

23.13.48 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 1-bit alpha case, finer scaling can be achieved by changing BLEND_WEIGHT* instead.

Offset: 715h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxx

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.

23.14 WINBUF_C Registers

The registers under DC_WINBUF are triple-buffered.

23.14.1 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 3-ii 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 or linear mode. However, for linear mode, the addresses calculated are real physical addresses, but for tile mode, these addresses will be translated to the real physical addresses by the memory controller client before used.

- When a window is host triggered, starting address of a window is calculated as follows by HW.
  - 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 HW.

- 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 HW.

    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 multiple of 2 in yuv planar format (the last bit is ignored), but with no restrictions on other color formats.
  - ADDR_V_OFFSET has no restrictions
- 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 formula in 2-i,ii 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 multiple of 2. The least significant bit of H_OFFSTE is ignored.

    For 32-bpp formats,

    (START_ADDR+H_OFFSTE) needs to be multiple of 4. The least two significant bits of H_OFFSTE are ignored.
  - 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 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 multiple of 4.

ADDR_H_OFFSET:

Needs to be a multiple of 2. If needs to point to odd pixel position, program ADDR_H_OFFSET to be the previous position (or the next position if H-flipped) and program H_INITIAL_DDA bit 12 to 1.

ADDR_V_OFFSET:

Needs to be a multiple of 2. If needs to point to odd line number, program ADDR_V_OFFSTE to be the previous line number (or next line number if V-flipped) and program V_INITIAL_DDA bit 12 to 1.

- 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, the frame start and frame end flag are active every time a buffer index is sent.

Offset: 800h | Read/Write: R/W | Reset: 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 LSB are not used for 32-bpp non-planar pixel format. For YUV planar pixel format, this specifies start address for the Y plane.

23.14.2 DC_WINBUF_C_START_ADDR_NS_0

Window C Shadowed Start Address

Offset: 801h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_START_ADDR_NS: Window C Shadowed Start Address. This is ARM set shadow of Start Address.

23.14.3 DC_WINBUF_C_START_ADDR_U_0

Window C Start Address for U plane

Offset: 802h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_START_ADDR_U: Window C Start Address for U plane. This is a byte address.

23.14.4 DC_WINBUF_C_START_ADDR_U_NS_0

Window C Shadowed Start Address for U plane

Offset: 803h | Read/Write: R/W | Reset: 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

23.14.5 DC_WINBUF_C_START_ADDR_V_0

Window C Start Address for V plane

Offset: 804h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_START_ADDR_V: Window C Start Address for V plane. This is a byte address.

23.14.6 DC_WINBUF_C_START_ADDR_V_NS_0

Window C Shadowed Start Address for V plane

Offset: 805h | Read/Write: R/W | Reset: 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

23.14.7 DC_WINBUF_C_ADDR_H_OFFSET_0

Window C Horizontal address offset

Offset: 806h | Read/Write: R/W | Reset: 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 LSB 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 HW.

23.14.8 DC_WINBUF_C_ADDR_H_OFFSET_NS_0

Window C Shadowed Horizontal address offset

Offset: 807h | Read/Write: R/W | Reset: 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

23.14.9 DC_WINBUF_C_ADDR_V_OFFSET_0

Window C Vertical address offset

Offset: 808h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	C_ADDR_V_OFFSET: Window C Vertical address offset. This is a byte address. The LSB is not used for 16-bpp non-planar pixel format and the last 2 LSB 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 HW.

23.14.10 DC_WINBUF_C_ADDR_V_OFFSET_NS_0

Window C Shadowed Vertical address offset

Offset: 809h | Read/Write: R/W | Reset: 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

23.14.11 DC_WINBUF_C_UFLOW_STATUS_0

Window C FIFO Underflow Status Register

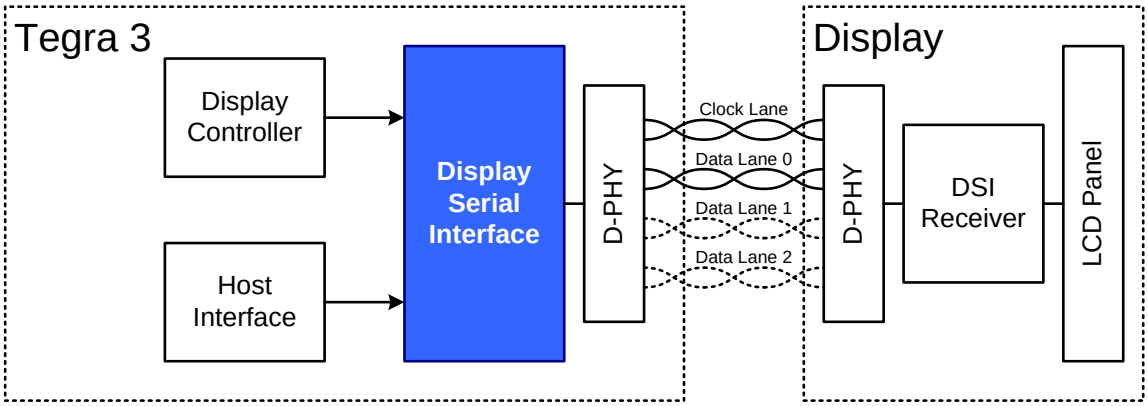
Offset: 80ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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.

24.0 MIPI-DSI (DISPLAY SERIAL INTERFACE)

The Display Serial Interface (DSI) is a MIPI standard serial bitstream, intended to provide a low pin count interface to a display panel. DSI reduces both package pin-count and I/O power consumption compared to parallel solutions. It transfers pixel data from either one of the display controllers (internal to Tegra® 3) 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 55.

Figure 55 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.

24.1 Clocking

- Host Clock = 166 MHz {Max frequency}
- Pixel Clock = 2.49 to 91.0 MHz for all modes.
- Application/Lane Management Byte Clock = function of lanes, pixel clock, pixel depth. 10 to 125 MHz.
- Fixed LP receive clock = 72 MHz
- Lane Bit clock = 40 to 500 MHz differential clock, DDR data (80 to 1000 bits/sec).

More details on pixel clock/byte_clock selection is explained in the next section.

24.1.1 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 50. As can be seen, for some modes, these ratios are far from being simple powers of 2.

Table 50. 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

Since the pixel clock is effectively derived from the D-PHY bit clock, it is perhaps 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 51. 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 2. However, if you do this, 18 bpp (packed) over 2 lanes also becomes a problem.

Table 51. 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
3	16 / 3	6	8	8
4	4	9 / 2	6	6

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. In the straightforward case where pixel clock is derived from PLLD, programming pixel clock divider using Table 51 above and PLLD programming will suffice clocking requirements. PLLD is covered in section 24.6.1 below.

24.2 Operating Modes

24.2.1 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:

24.2.1.1 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, 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.

24.2.1.2 Non-Burst Mode

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.

24.2.1.3 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.

24.2.2 Command Mode

Communication with the peripheral is by asynchronously timed and variable length packets. The packets may contain video data 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.

24.3 Display Controller Interface

The interface between the display controller and the DSI unit consists of the following elements:

- 24-bit wide pixel bus with an associated valid signal. The valid will only be high during the active portion of a video line.
- 1-bit Horizontal Sync (HS) signal that will toggle once per line.
- 3-bit Lin-Type code. These bits should remain stable for at least 2 pixel clock periods prior to the Horizontal Sync signal changing state. This is to ensure that on the DSI side, the state of these 3 bits is correctly captured.

The DSI unit uses these signals in the following manner: When the display controller toggles the HS signal, the DSI captures the state of the line type code. 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. Once the packet sequence has been selected, the DSI sends these packets into its internal packet FIFO. When a packet that requires data from the display controller is encountered, the DSI pauses the packet writing operation, until data arrives from the display controller on the pixel bus. When the data arrives – indicated by the assertion of the valid signals – the data is written into the FIFO. This process continues until all the packets for that line type have been written.

Separately, when the HS signal arrives, a counter is started. When the counter reaches a pre-programmed value, the logic on the read side of the DSI internal FIFO is instructed to start pulling packets out of the FIFO and transmit them over the DSI link to the display device.

24.4 Host Interface

Not all displays can accept a raster-based packet stream. For some low-bandwidth displays with small pixel and line dimensions, it is more efficient to have a frame store in the display device itself and to only update the contents of the frame store with occasional packet sequences. For these types of display, there is a SW accessible interface for sending command packets using 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 SW 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.

24.5 FIFO Buffers

24.5.1 Video Mode Operation

24.5.1.1 Writing Data

The following packets are written into the large data FIFO based on the various triggering events described.

Table 52 FIFO Trigger Events

Trigger Event	Packet
Leading edge of Vertical Sync.	V Sync Start
Trailing edge of Vertical Sync	V Sync End
Leading edge of Horizontal Sync, *unless* simultaneous with leading or trailing edge of vertical sync	H Sync Start
Trailing edge of Horizontal Sync	H Sync 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)

24.5.1.2 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.

24.5.2 Command Mode Operation

24.5.2.1 Writing Data from the Display Controller

Upon receipt of the leading edge of Vertical Sync, the display 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, but it is the responsibility of SW 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.

24.5.2.2 Writing Data from the Host

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, as this is performed by the hardware on the other side of the data FIFO.

24.5.2.3 Reading Data

The initiation of the reading of data from the FIFO is triggered by a FIFO threshold. Once the threshold is reached, the FIFO starts to be drained by the D-PHY 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, nor FIFO underflow caused by a lack of sufficient data in the FIFO.

24.5.3 FIFO Sizing

FIFO Threshold Video Non-bust modes is calculated as

Threshold = (Length x OutRate) / (OutRate + InRate)

Where,

- InRate = pixClk
- OutRate = (ByteClk x NumOfLanes) / BytePerPix
- Length = input frame width
- Threshold = minimum fifo depth to start transfer data to CIL

Table 53 Output Rates and Thresholds for Width=800, ByteClk=128, InRate=44.

		Number of Lanes			
Bytes per pixel		1	2	3	4
2	Out Rate	64	128	192	256
2	Threshold	474	595	650	682
2.25	Out Rate	56	113	170	227
2.25	Threshold	448	575	635	670
3	Out Rate	42	85	128	170
3	Threshold	390	527	595	635

Table 54 Output Rates and Thresholds for Width=1024, ByteClk=128 InRate=44.

		Number of Lanes			
Bytes per pixel		1	2	3	4
2	Out Rate	64	128	192	256
2	Threshold	606	762	833	873
2.25	Out Rate	56	113	170	227
2.25	Threshold	573	737	813	857
3	Out Rate	42	85	128	170
3	Threshold	500	674	762	813

24.6 Programming Guidelines

The packet timing diagrams in this section use the following key:

Figure 56 Video Mode Timing Diagram Key.

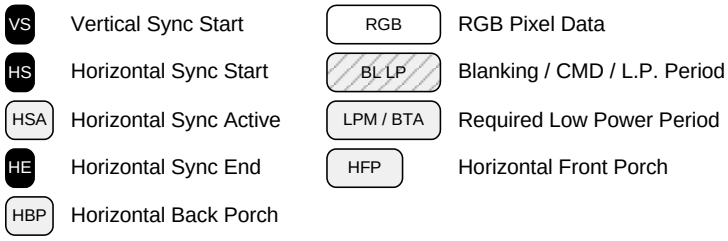


Table 55 Timing Diagram parameter list

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

Table 56 DSI Register Overview

Register	Description
DSI_RD_DATA	BTA read-return FIFO output
DSI_WR_DATA	Host Transmit FIFO input
DSI_POWER_CONTROL	
INT_ENABLE	Interrupt Enables
INT_STATUS	Interrupt Status
INT_MASK	Interrupt Mask
HOST_DSI_CONTROL	Host-specific DSI controls
DSI_CONTROL	General and Video DSI controls
DSI_SOL_DELAY	Start-Of-Line delay register
DSI_MAX_THRESHOLD	FIFO transmit start threshold
DSI_TRIGGER	Manual transmit trigger register
DSI_TX_CRC	Verification transmitted data CRC
DSI_INIT_SEQ_CONTROL	Initialization sequence control
DSI_INIT_SEQ_DATA_0	Initialization data 0
DSI_INIT_SEQ_DATA_1	Initialization data 1
DSI_INIT_SEQ_DATA_2	Initialization data 2
DSI_INIT_SEQ_DATA_3	Initialization data 3
DSI_PKT_SEQ_[0..5]_LO	Packet Sequence [0..5] (low)
DSI_PKT_SEQ_[0..5]_HI	Packet Sequence [0..5] (high)
DSI_DCS_CMDS	DCS commands for Line Type 3 and 5
DSI_PKT_LEN_0_1	Packet Lengths for packets 0 and 1
DSI_PKT_LEN_2_3	Packet Lengths for packets 2 and 3

Register	Description
DSI_PKT_LEN_4_5	Packet Lengths for packets 4 and 5
DSI_PKT_LEN_6_7	Packet Lengths for packets 6 and 7
DSI_PHY_TIMING_0	Low-level D-PHY timing control
DSI_PHY_TIMING_1	Low-level D-PHY timing control
DSI_BTA_TIMING	Low-level D-PHY timing control
DSI_TIMEOUT_0	HTX_TO and LRXH_TO durations
DSI_TIMEOUT_1	TA_TO and PR_TO durations
DSI_TO_TALLY	Count of how many TOs have occurred

24.6.1 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.

24.6.1.1 PLLD Programming

The DSI PLL is called PLLD in the Tegra 3 Devices. Just like the other PLLs in the system, PLLD has an M, N and P register for programming the various divide ratios for the clock. When used to drive 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 actually 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);
}
```

24.6.1.2 Video Mode Start-up

All 3 video modes – Burst, Non-Burst and Non-Burst with Sync Ends – essentially 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, 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`. Select which Display Controller will source the video data.
- Configure PHY timing registers and timeout registers
- Enable 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.

24.6.1.3 Command Mode start-up

Set up the clocks. Program and enable the DSI PLL (PLLD). In addition, 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 PLLD output clock frequency is essentially arbitrary. The suitability of 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, 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.
- Configure PHY timing registers and timeout registers
- Enable DSI.

The DSI should now be ready to accept Command Packets written to the `DSI_WR_DATA` register.

24.6.2 Video Mode Programming

Packet sequences for video mode are somewhat complex and there is on-going debate within the MIPI DSI workgroup regarding the precise sequences and whether or not certain packets are optional. In order to make the hardware flexible enough to cope with all current video mode sequences and any possible future sequences or sequence variations, a more-or-less open-ended programming model has been adopted.

The programming model is as follows:

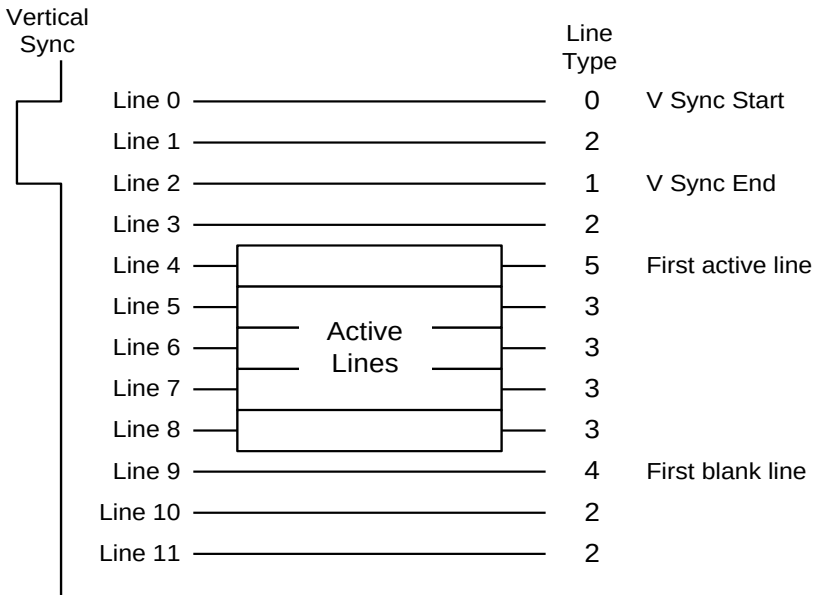
Each line of video output will have associated with it a “line type”. This line type is automatically generated by the display controller HW according to the following table:

Table 57 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 V sync 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 57 illustrates a typical raster showing the relationship between various events in the raster and the line types.

Figure 57 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 58 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 59 shows how all six packet descriptors fit into the pair of 32-bit packet sequence registers.

Table 59 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. The reason an indirect pointer is used, rather than storing the packet length directly in the packet descriptor is that the packet lengths are physically large – 16 bits – but there is a lot of re-use. There are actually 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.

It should be noted 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 implies this packet is 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 HW 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 60. 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. This open-ended possibility is very powerful, but may lead to confusion. It is therefore recommended that the suggested packet length assignment for various length registers is followed. This consistency will hopefully reduce confusion when debugging packet descriptors and packet sequences.

Table 60 Packet Length registers and assignments

Register	Field	Suggested Assignment
DSI_PKT_LEN_0_1	LENGTH_0	Must be used for short packets
	LENGTH_1	H sync active length (HSA)

Register	Field	Suggested Assignment
DSI_PKT_LEN_2_3	LENGTH_2	H back porch length (HBP)
	LENGTH_3	H active length (ACTIVE)
DSI_PKT_LEN_4_5	LENGTH_4	H front porch length (HFP)
	LENGTH_5	- reserved -
DSI_PKT_LEN_6_7	LENGTH_6	- reserved -
	LENGTH_7	- reserved -

When programming the length registers, it should be remembered that these are 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 is given in the examples that follow. The identification of the length given in the tables and equations has been included in braces in Table 60 so that the values in the examples can be easily tied to the registers.

24.6.3 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 display. This is especially true of Burst-Mode, since 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 is used to set this delay.

24.6.3.1 None-Burst Mode

In non-burst mode, the rate at which 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 on the order of 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. Please 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:

$$SOL\_DELAY = 8 * F_{DSI} / F_{pixel}$$

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.

24.6.3.2 Burst Mode

Burst mode is much more complex because 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 H sync) and the H active time. So, for Burst Mode:

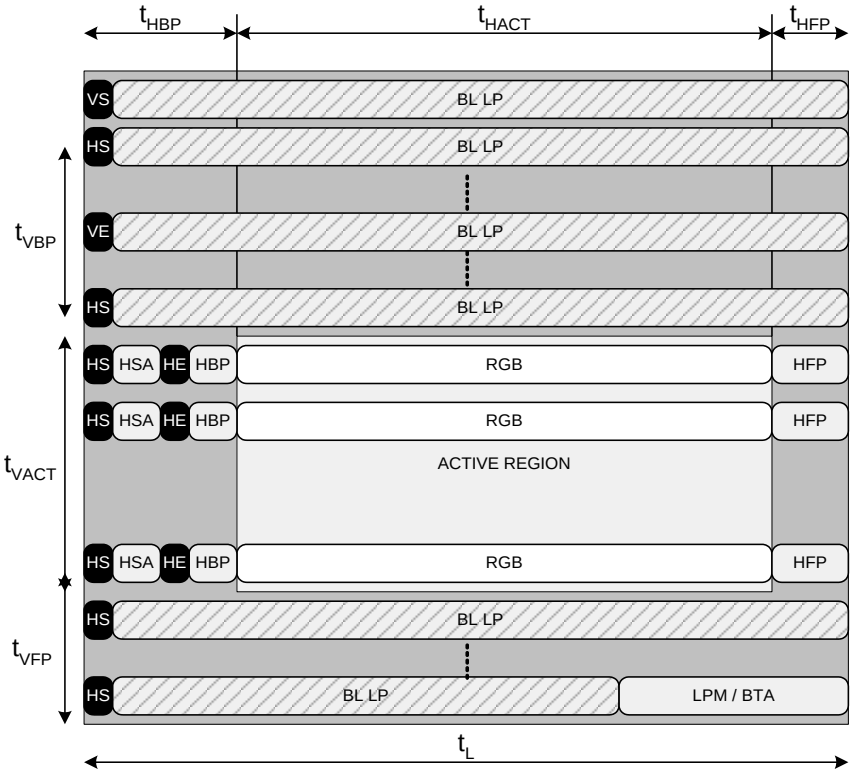
$$SOL\_DELAY = ((T_{HBP} + T_{active}) * F_{DSI} / F_{pixel}) – (T_{active} * F_{DSI\_NB} / F_{pixel\_NB})$$

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.

24.6.4 Non-Burst Mode with Start and End

In this mode, an attempt is made to accurately convey traditional raster synchronization information across the link to the peripheral. This is achieved by sending both start and end sync packets.

Figure 58 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. There is some debate going on as to whether the HFP blank packet is required, or even desirable. This may get replaced by a period of Blanking / Low Power mode.

The detail of how a single line is constructed from multiple packets is shown in the figure below. In addition, it is necessary for the packet constructor on the display clock side of the display / DSI boundary FIFO to calculate the packet size for four different packets on the line.

It is also suggested that there is a running-disparity adjustment made to the packet sizes so that the average pixel rate on the DSI side matches the display output rate.

Figure 59 Non-Burst Mode with Start and End Packet Timing Detail

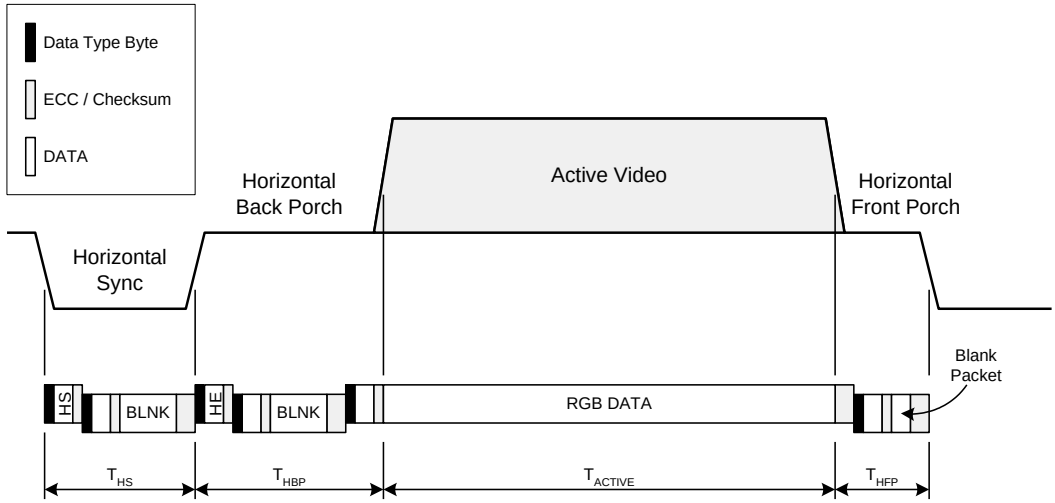


Table 61 Payload Size Table - Non-Burst Mode with Start and End

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.

N = Number of lanes used

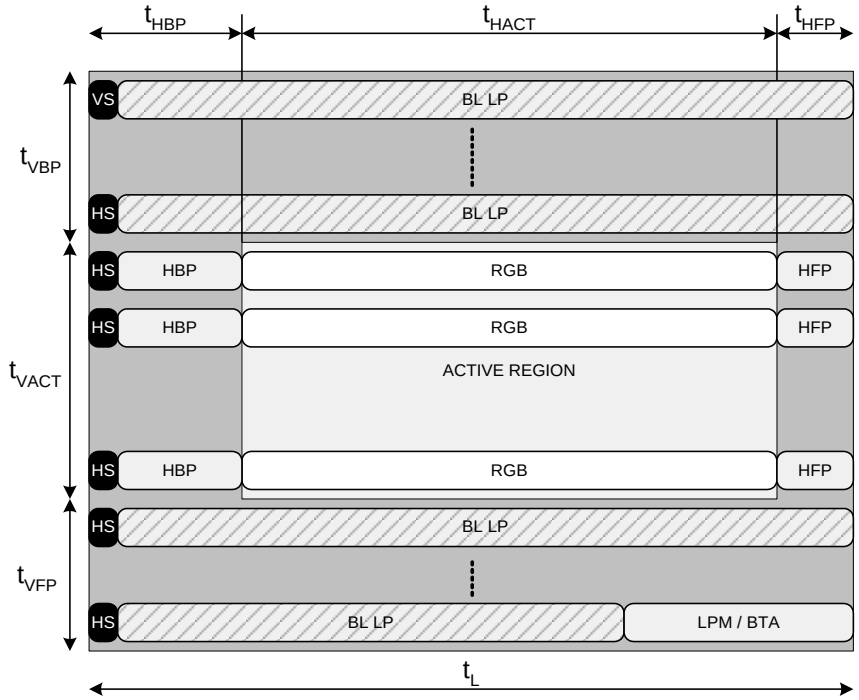
Table 62 Line Type Packet Sequences - Non-Burst with Sync Ends

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	0								
1	VE	0	EOT	0								
2	HS	0	EOT	0								
3	HS	0	BLNK	1	HE	0	BLNK	2	RGB	3	BLNK	4
4	HS	0	EOT	0								
5	HS	0	BLNK	1	HE	0	BLNK	2	RGB	3	BLNK	4

24.6.5 Non-Burst Mode (Without Ends)

This mode relaxes the requirement to mimic the generation of sync pulses and instead 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 60 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. As of writing, there is some debate as to whether the HFP blank packet is required, or even desirable. This may get replaced by a period of Blanking / Low Power mode.

The detail of how a single line is constructed from multiple packets is shown in the figure below. In addition, it will be necessary for the packet constructor on the display clock side of the display / DSI boundary FIFO to calculate the packet size for three different packets on the line. It is also suggested that there is a running-disparity adjustment made to the packet sizes so that the average pixel rate on the DSI side matches the display output rate.

Figure 61 Non-Burst Mode Packet Timing Detail

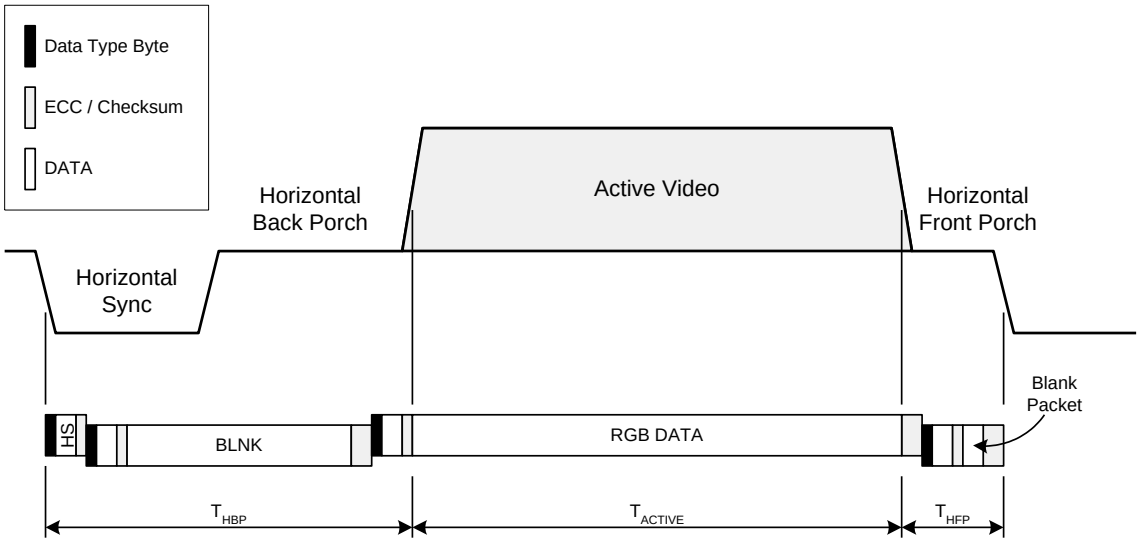


Table 63 Payload Size Table - Non-Burst Mode

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.

N = Number of Lanes used.

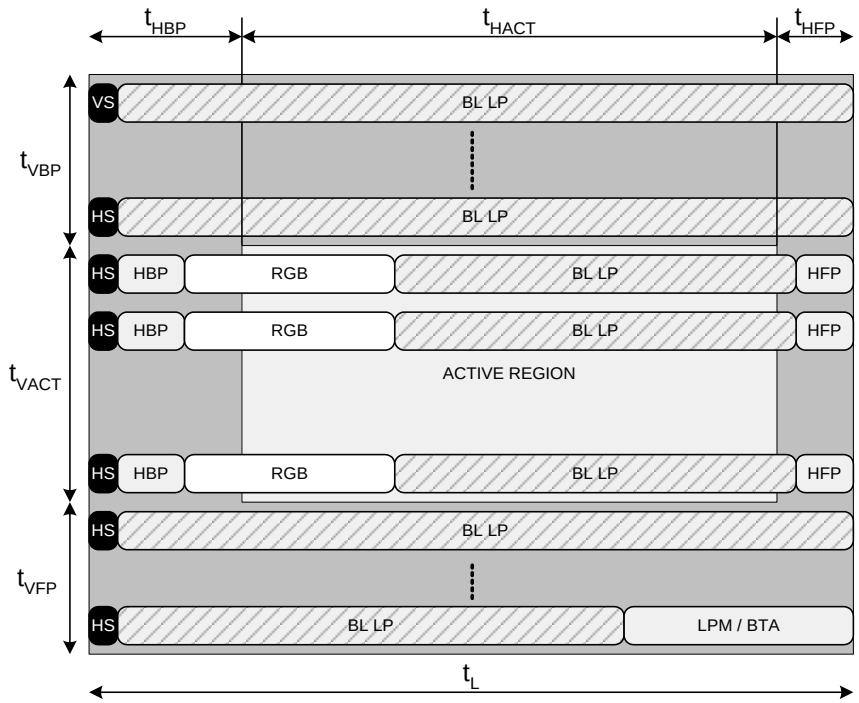
Table 64 Line Type Packet Sequences – Non Burst

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	VS	0	EOT	0								
1	HS	0	EOT	0								
2	HS	0	EOT	0								
3	HS	0	BLNK	2	RGB	3	BLNK	4				
4	HS	0	EOT	0								
5	HS	0	BLNK	2	RGB	3	BLNK	4				

24.6.6 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 each line that can be used for the transmission of other packets.

Figure 62 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 essentially simultaneously with the video stream. The NULL and HFP packets may be optional. Check the latest MIPI DSI specification for clarification.

Table 65 Line Type Packet Sequences - Bust Mode

LT	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len	ID	Len
0	NULL	4	VS	0	EOT	0						
1	NULL	4	HS	0	EOT	0						
2	NULL	4	HS	0	EOT	0						
3	BLNK	4	HS	0	BLNK	2	RGB	3	EOT	0		
4	NULL	4	HS	0	EOT	0						
5	BLNK	4	HS	0	BLNK	2	RGB	3	EOT	0		

24.6.7 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.

24.6.7.1 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 (ECC) and Cyclic Redundancy Checks (CRC), but all other data is passed un-altered to the DSI physical layer.

Host Packet Writes

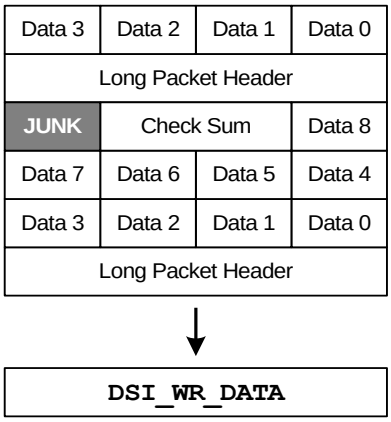
Packets are written to the HW by writing to the `DSI_WR_DATA` register. The data written is passed into the DSI Host transmit FIFO. If the data is a packet header and the `ECC_ENABLE` field in the `HOST_DSI_CONTROL` register is set to `ENABLE`, then the MSBs of the 32 bit packet header word is replaced 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 HW 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. 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 about the last packet, but should re-align with the register. See Figure 63 for an example.

Figure 63 Packet alignment for writes to DSI_WR_DATA



Host Packet Transmission

Control of when the Host write FIFO starts to drain (flush) can be programmed to come from several sources:

- Start Of Line signal from the Display Controller
- Once the FIFO has some programmable quantity of data in it.
- Explicit FLUSH bit in a control register

Start of Line (HOST_TX_TRIG_SRC == SOL **)**

This mode would typically be used when the Host is trying to emulate Video Mode to drive a raster-based display. Currently this mode of operation is a P1 feature.

Threshold (HOST_TX_TRIG_SRC == FIFO_LEVEL**)**

The FIFO can be programmed to automatically start transmission once a certain amount of data has been written to the FIFO. This is useful if the amount of data to be transmitted is already known and is essentially constant, or at least consistently more than the threshold amount. Software can then simply write to the `DSI_WR_DATA` register and transmission will start automatically. However, it should be ensured that if the threshold is set such that not all the data for a packet will have been written when the threshold is met, that software will write all the required data before the transmission has drained the FIFO completely.

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 it recommended that all data required for transmission is written to the `DSI_WR_DATA` register prior to setting this bit.

24.6.7.2 Command Mode from Display Controller

This mode of operation is intended to allow 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. Of course, the data ***will*** be sent in an isochronous manner since it will be coming from the display controller, but the commands sent will be DCS control commands, rather than Video Mode timing and blanking packets.

Setup

There are four steps to operating in this mode:

1. Program the Peripheral display device using DCS commands via the Host Command interface. It is especially important to send the set_column_address, set_page_address and set_pixel_format commands. These effectively define the area to which we will be writing pixels.
2. Program the DSI_INIT_SEQ_CONTROL and DSI_INIT_SEQ_DATA_* registers in the DSI interface with appropriate values. Any commands required to be sent once per frame and *every* frame by way of set-up 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 below on pixel format restrictions.
4. Program the DSI_PKT_SEQ_* registers. Program the registers in the flowing way:
    a. 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.
    b. Packet Sequence registers for Line Type 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 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 66. Unfortunately, due to limitations in the current design of the DSI hardware, only 3 of the 6 formats can be supported.

The formats supported are also shown in the “supported” column of Table 66. Do not set BIT18NP as a pixel format. There is no DCS equivalent of this format, so undefined behavior may result if this format is set.

Table 66 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. To this end, Line Type 4 (First blank line) is reserved to indicate to the HW when it should attempt to send any DCS command packets requested by SW.

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 HW 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.

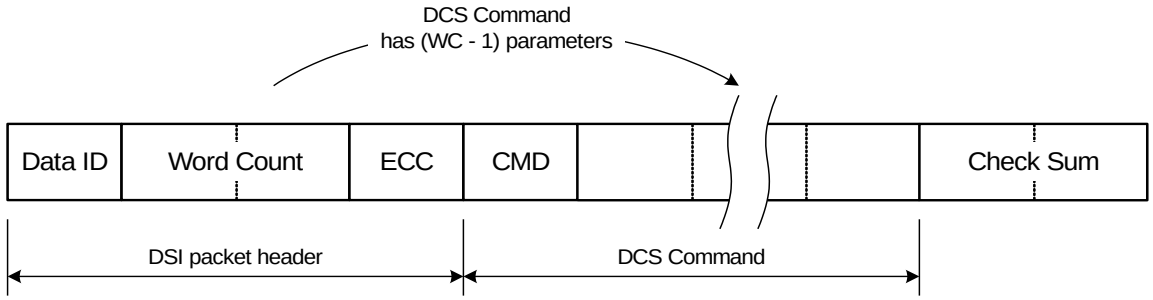
Note: There will be no BTA permitted in this mode, so no ACK, ACK with error report or read return data will be generated.

24.6.8 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. A diagram of how a DCS Long Write packet is constructed is shown below in the next figure.

Figure 64 DCS command Placement in DSI Long Packet



24.6.9 Function Programming

24.6.9.1 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 [2]. However, this is 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 actually 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.

24.6.9.2 CRC Insertion

The DSI check sum 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, it is easier to 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.

24.6.10 Read Data Return

DSI is a bi-directional interface. Data is returned from the peripheral display device only after the display controller has requested information by issuing a Bust Turn Around (BTA). All returned data is written into a FIFO that can be read using Host reads of a DSI register.

24.6.10.1 Reading Peripheral Registers

A typical application of BTA would be 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.

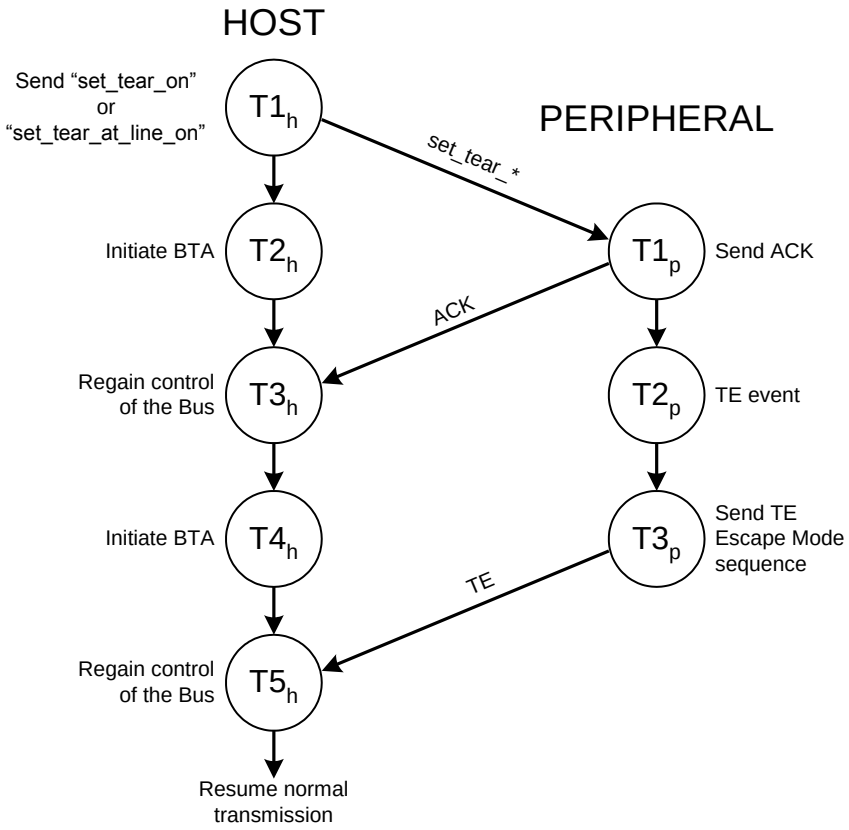
24.6.10.2 Bus Turn Around

- 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. Obviously, the length of long packets must be severely restricted.
- There will be no ECC or CS check performed by hardware on the read return data. Entire packets will simply be made available to software to perform whatever checks they desire.
- There will be 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.

24.6.11 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 65. 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 $T1_h$.
2. Wait for ACK to come back from the peripheral. This is states $T2_h$ and $T3_h$.
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 $T4_h$.
4. Wait for TE return byte from the peripheral. This is state $T5_h$.

24.6.12 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.

24.6.12.1 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 67.

Table 67 Peripheral response to various conditions

Condition	Non-Read Packet	Read Packet
No error	ACK	Read Data
Corrected single bit error	ACK w/ERR	Read Data + ACK w/ERR
Non-corrected multi-bit error	ACK w/ERR	ACK w/ERR
SOT, EOT, VI ID or other D-PHY error	ACK w/ERR	ACK w/ERR

It is the responsibility of software to read this data from the packet returned in the packet return FIFO and process as required. No action will be taken by the hardware based on the error report returned in the ACK packet.

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 is a single byte trigger and 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 68 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
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

24.6.13 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 69 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 69, 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 SW to determine when the peripheral has been reset.

Table 70 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
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

24.7 MIPI-DSI Registers

24.7.1 DSI_INCR_SYNCPT_0

Offset: 000h | Read/Write: R/W | Reset: 0b0000000000000000

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

24.7.2 DSI_INCR_SYNCPT_CNTRL_0

Offset: 001h | Read/Write: R/W | Reset: 0b0xxxxxxx0

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.

24.7.3 DSI_INCR_SYNCPT_ERROR_0

Offset: 002h | Read/Write: R/W | Reset: 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.

24.7.4 DSI_CTXSW_0

Context switch register (common to all modules). Includes the current channel/class (which is writable by SW) 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). SW 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: 008h | Read/Write: R/W | Reset: 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.

24.7.5 DSI_DSI_RD_DATA_0

DSI read return data

Offset: 009h | Read/Write: RO | Reset: 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. FIFO has NV_DSI_HOST_DATA_RETURN_FIFO_DEPTH entries.

24.7.6 DSI_DSI_WR_DATA_0

Host FIFO write input

Offset: 00ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_WR_DATA: Each write to this register will puch 32 bits into the 32 bit wide Host data FIFO. FIFO has NV_DSI_HOST_DATA_FIFO_DEPTH entries

24.7.7 DSI_DSI_POWER_CONTROL_0

Display Power Control. DSI Enable

Offset: 00bh | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	LEG_DSI_ENABLE: DSI interface Enable 0 = DISABLE : Disable DSI 1 = ENABLE : Enable DSI

24.7.8 DSI_INT_ENABLE_0

Interrupt enable register

Offset: 00ch | Read/Write: R/W | Reset: 0b1

Bit	Reset	Description
0	0x1	CTXSW_INT_ENABLE: Context Switch Interrupt Enable 0 = DISABLE : interrupt disabled 1 = ENABLE : interrupt enabled

24.7.9 DSI_INT_STATUS_0

Interrupt Status register

Offset: 00dh | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	CTXSW_INT: Context switch interrupt status (clear on write)

24.7.10 DSI_INT_MASK_0

Setting bits in this register masked 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.

Interrupt Mask

Offset: 00eh | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	CTXSW_INT_MASK: Context Switch Interrupt Mask 0 = MASKED : interrupt masked 1 = NOTMASKED : interrupt not masked

24.7.11 DSI_HOST_DSI_CONTROL_0

DSI control register when input is from HOST

Offset: 00fh | Read/Write: R/W | Reset: 0b00x000xx00xx0000000011

Bit	Reset	Description
21	0x0	FIFO_STAT_RESET: write only bit to clear FIFO underflow/overflow flags. If there new underflow/overflow event occurs during the same time, current access cant 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 1 = ENABLE Causes an Escape Mode Command to be sent to reset the 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. HW clears this bit upon completion of issuing "Trigger Reset" OR called "Reset Entry" command is sent out.
6	0x0	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 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.
3	0x0	IMM_BTA: Generate BTA immediately, eg. 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 HW generation of CS (Host must calculate CS). 1 = ENABLE : enable HW 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 HW generation of ECC (Host must calculate ECC). 1 = ENABLE : enable HW generation of ECC.

24.7.12 DSI_DSI_CONTROL_0

General DSI Control register

Offset: 010h | Read/Write: R/W | Reset: 0b0xxxxxxxxxx0xx00xx00xx00xx000000

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 toggles. Also TX CRC computation aid will turn on
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 Virtual channel is sent as part of 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 HW. 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. (NOTE: Currently, this is illegal) 3 = FOUR : 4 data lanes. (NOTE: Currently, this is illegal)
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

24.7.13 DSI_DSI_SOL_DELAY_0

Number of byte-clock counts to wait after reception of.

Offset: 011h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	SOL_DELAY: Start Of Line before generating output packets.

24.7.14 DSI_DSI_MAX_THRESHOLD_0

Maximum threshold registers for DSI related packets.

Offset: 012h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

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.

24.7.15 DSI_DSI_TRIGGER_0

Manual transmissions trigger register.

Offset: 013h | Read/Write: R/W | Reset: 0bxx

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 HW auto clears on operation completion

Bit	Reset	Description
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 HW auto clears on operation completion

24.7.16 DSI_DSI_TX_CRC_0

Transmission CRC

Offset: 014h | Read/Write: RO | Reset: 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 HW 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 SW to read the current result.

24.7.17 DSI_DSI_STATUS_0

DSI Status register

Offset: 015h | Read/Write: RO | Reset: 0bxxxxxxxxxxx

Bit	Reset	Description
10	x	DSI_IDLE: Indicated that DSI is IDLE
9	X	LB_UNDERFLOW: Indicates that Line buffer underflow event happened
8	X	LB_OVERFLOW: Indicates that 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, SW 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 SW. However, under error conditions, insufficient data may have been read from the FIFO. SW should therefore check this field to make sure it is 0 after reading all the information it expected to get.

24.8 Initialization Sequence Registers

24.8.1 DSI_DSI_INIT_SEQ_CONTROL_0

DSI Initialization Sequence Control.

Offset: 01ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx0

Bit	Reset	Description
13:8	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 32 Invalid programmable values: 33 to 63

Bit	Reset	Description
0	0x0	DSI_SEND_INIT_SEQUENCE: Send Initialization Sequence (IS) 0 = DISABLE : 1 = ENABLE :

24.8.2 DSI_DSI_INIT_SEQ_DATA_0_0

DSI Init Sequence Write Data 0.

Offset: 01bh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_0: DSI Init Sequence Write Data bits 31-0

24.8.3 DSI_DSI_INIT_SEQ_DATA_1_0

DSI Init Sequence Write Data 1.

Offset: 01ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_1: DSI Init Sequence Write Data bits 31-0

24.8.4 DSI_DSI_INIT_SEQ_DATA_2_0

DSI Init Sequence Write Data 2.

Offset: 01dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_2: DSI Init Sequence Write Data bits 31-0

24.8.5 DSI_DSI_INIT_SEQ_DATA_3_0

DSI Init Sequence Write Data 3.

Offset: 01eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_3: DSI Init Sequence Write Data bits 31-0

24.8.6 DSI_DSI_INIT_SEQ_DATA_4_0

DSI Init Sequence Write Data 4

Offset: 01fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_4: DSI Init Sequence Write Data bits 31-0

24.8.7 DSI_DSI_INIT_SEQ_DATA_5_0

DSI Init Sequence Write Data 5

Offset: 020h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_5: DSI Init Sequence Write Data bits 31-0

24.8.8 DSI_DSI_INIT_SEQ_DATA_6_0

DSI Init Sequence Write Data 6

Offset: 021h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_6: DSI Init Sequence Write Data bits 31-0

24.8.9 DSI_DSI_INIT_SEQ_DATA_7_0

DSI Init Sequence Write Data 7

Offset: 022h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DSI_INIT_SEQ_DATA_7: DSI Init Sequence Write Data bits 31-0

24.9 Packet Sequence Registers

24.9.1 DSI_DSI_PKT_SEQ_0_LO_0

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 second digit denotes the packet number within each sequence. For example, field PKT_23_ID, defines the ID for packet 3 in sequence 2.

DSI Packet Sequence 0 LO half

Offset: 023h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

Bit	Reset	Description
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

24.9.2 DSI_DSI_PKT_SEQ_0_HI_0

DSI Packet Sequence 0 HI half

Note: 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: 024h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

24.9.3 DSI_DSI_PKT_SEQ_1_LO_0

DSI Packet Sequence 1 LO half

> **Note:** 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: 025h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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
2:0	X	PKT_10_SIZE: Packet 0 size pointer

24.9.4 DSI_DSI_PKT_SEQ_1_HI_0

DSI Packet Sequence 1 HI half

Offset: 026h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

Bit	Reset	Description
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

24.9.5 DSI_DSI_PKT_SEQ_2_LO_0

DSI Packet Sequence 2 LO half

Note: 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: 027h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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
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

24.9.6 DSI_DSI_PKT_SEQ_2_HI_0

DSI Packet Sequence 2 HI half

Offset: 028h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

24.9.7 DSI_DSI_PKT_SEQ_3_LO_0

DSI Packet Sequence 3 LO half

Note: 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: 029h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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
18:13	X	PKT_31_ID: Packet 1 Packet ID

Bit	Reset	Description
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

24.9.8 DSI_DSI_PKT_SEQ_3_HI_0

DSI Packet Sequence 3 HI half

Offset: 02ah | Read/Write: R/W | Reset: 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

24.9.9 DSI_DSI_PKT_SEQ_4_LO_0

DSI Packet Sequence 4 LO half

> **Note:** 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: 02bh | Read/Write: R/W | Reset: 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
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

24.9.10 DSI_DSI_PKT_SEQ_4_HI_0

DSI Packet Sequence 4 HI half

Offset: 02ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

Bit	Reset	Description
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

24.9.11 DSI_DSI_PKT_SEQ_5_LO_0

DSI Packet Sequence 4 LO half

Note: Line Type 5 is associated with the first active line and 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: 02dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
30	X	SEQ_5_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_52_EN: Packet 2 enable 0 = DISABLE 1 = ENABLE
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

24.9.12 DSI_DSI_PKT_SEQ_5_HI_0

DSI Packet Sequence 5 HI half

Offset: 02eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

24.9.13 DSI_DSI_DCS_CMDS_0

Note: DCS COMMAND AND PACKET LENGTH REGISTERS

DCS command IDs used for Line Type 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: 033h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

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.

24.9.14 DSI_DSI_PKT_LEN_0_1_0

DSI packet lengths 0 and 1

Offset: 034h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	LENGTH_1: Packet length 1 (in bytes)
15:0	X	LENGTH_0: Packet length 0 (in bytes)

24.9.15 DSI_DSI_PKT_LEN_2_3_0

DSI packet lengths 2 and 3

Offset: 035h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	LENGTH_3: Packet length 3 (in bytes)
15:0	X	LENGTH_2: Packet length 2 (in bytes)

24.9.16 DSI_DSI_PKT_LEN_4_5_0

DSI packet lengths 4 and 5

Offset: 036h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	LENGTH_5: Packet length 5 (in bytes)
15:0	X	LENGTH_4: Packet length 4 (in bytes)

24.9.17 DSI_DSI_PKT_LEN_6_7_0

DSI packet lengths 6 and 7

Offset: 037h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	LENGTH_7: Packet length 7 (in bytes)
15:0	X	LENGTH_6: Packet length 6 (in bytes)

24.10 Physical Interface Timing Registers

24.10.1 DSI_DSI_PHY_TIMING_0_0

DSI D-PHY timing register 0

Offset: 03ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	DSI_THSDEXIT: Time to drive LP11 after HS
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

24.10.2 DSI_DSI_PHY_TIMING_1_0

DSI D-PHY timing register 1

Offset: 03dh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	DSI_TCLKTRAIL: Time to drive HS0 before clock goes to LP1 1
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

24.10.3 DSI_DSI_PHY_TIMING_2_0

DSI D-PHY timing register 2

Offset: 03eh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
23:16	X	DSI_TCLKPREPARE: Time to drive LP0 before CLK_ZERO starts off on clock lane
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.

24.10.4 DSI_DSI_BTA_TIMING_0

DSI D-PHY Bus-Turn-Around timing

Offset: 03fh | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
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)

24.10.5 DSI_DSI_TIMEOUT_0_0

Note: 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 MIPI DSI Spec. section 7.2.2

DSI Time out terminal count register 0

Offset: 044h | Read/Write: R/W | Reset: 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

24.10.6 DSI_DSI_TIMEOUT_1_0

DSI Time out terminal count register 1

Offset: 045h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	PR_TO: Peripheral Reset duration.
15:0	X	TA_TO: Turn Around Time Out terminal count

24.10.7 DSI_DSI_TO_TALLY_0

Each time one of the time out counters reaches its terminal count, it will increment the associated tally register.

DSI Time out tally register

Offset: 046h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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

24.11 Physical Pad Control Registers

24.11.1 DSI_PAD_CONTROL_0

SW can program this register to work with different panels.

DSI PHY configuration register

Offset: 04bh | Read/Write: R/W | Reset: 0b00010000x00001110000x000x0000000

Bit	Reset	Description
31:30	0x0	DSI_PAD_HS_BSO: 1= enables BIAS and power regulators on for HS mode
29	0x0	DSI_PAD_HS_BSO_CLK: 1= enables BIAS and power regulators on for HS mode

Bit	Reset	Description
28	0x1	DSI_PAD_PULLDN_ENAB: 1= enable pad pulldn on power on
27	0x0	DSI_PAD_REV_CLK – Reverse clock polarity.
26:24	0x0	DSI_PAD_SLEWUPADJ – Pull up slew rate adjust. 0 = init 000 > 011 = slew rate increases 100 = same as 000 100 > 111 = slew rate decreases
22:20	0x0	DSI_PAD_SLEWDNADJ – Pull down slew rate adjust. 0 = init 000 > 011 = slew rate increases 100 = same as 000 100 > 111 = slew rate decreases
19	0x0	DSI_PAD_PREEMP_EN – HS driver pre-emphasis enable. 0 = init 1 = pre-emphasis enabled
18	0x1	DSI_PAD_PDIO_CLK – Power down for clock bit, drivers, receivers and contention detectors. 1 = init
17:16	0x3	DSI_PAD_PDIO – Power down for data bit,drivers,receivers and contention detectors. 3 = init
15:14	0x0	DSI_PAD_LPUPADJ – Driver pull up impedance control. 0 = init 00 = 130ohm (default) 01 = 110ohm 10 = 130ohm (same as 00) 11 = 150ohm
13:12	0x0	DSI_PAD_LPDNADJ – Driver pull down impedance control. 0 = init 00 = 130ohm (default) 01 = 110ohm 10 = 130ohm (same as 00) 11 = 150ohm
10:8	0x0	DSI_PAD_OUTADJCLK – Output trimmer delay for clock bit. Each tap delays 40ps. 0 = init
6:4	0x0	DSI_PAD_OUTADJ1 – Input delay trimmer for data bit 1. Each tap delays 40ps. 0 = init
3	0x0	DSI_PAD_BANDWD_IN – Increase bandwidth of differential receiver. 0 = init
2:0	0x0	DSI_PAD_OUTADJ0 – Input delay trimmer for data bit 0. Each tap delays 40ps. 0 = init

24.11.2 DSI_PAD_CONTROL_CD_0

Contention detection logic enable signals

Offset: 04ch | Read/Write: R/W | Reset: 0b000x000x000

Bit	Reset	Description
10:9	0x0	DSI_PAD_LP_BSO: 1= enables BIAS and power regulators on for LP mode
8	0x0	DSI_PAD_LP_BSO_CLK: 1= enables BIAS and power regulators on for LP mode
6:4	0x0	DSI_PAD_CDDNADJ: : Level adjust on low limit of detectionTie to 0 (DSI emu, miniTMC control)000->0.3v001->0.375v010->0.45v011->0.525v100->0.3v101->0.225v110->0.15v111->0.075v
2	0x0	DSI_PAD_CD_EN_CLK: Clock bit contention detector enable 1 = Enable
1:0	0x0	DSI_PAD_CD_EN: data bits contention detector enable 1 = EnableCD_EN_[1] = 0 in DSI

24.11.3 DSI_PAD_CD_STATUS_0

Contention detection status from MIPI PAD

Offset: 04dh | Read/Write: RO | Reset: 0bxxxxxx

Bit	Reset	Description
5	X	DSI_PAD_CDN_CLK
4	X	DSI_PAD_CDP_CLK
3:2	X	DSI_PAD_CDN
1:0	X	DSI_PAD_CDP

24.11.4 DSI_DSI_VID_MODE_CONTROL_0

Host Command Packet during Video Mode

Offset: 04eh | Read/Write: R/W | Reset: 0x00000000 (0b0000)

Bit	Reset	Description
3:1	0x0	DSI_LINE_TYPE: LINE TYPE on which host command packet to be transmitted 0 = ZERO : line type 0 1 = ONE : line type 1 2 = TWO : line type 2 3 = THREE : NA: host command packets on linetype 3 are invalid 4 = FOUR : line type 4 5 = FIVE : NA: host command packets on linetype 5 are invalid 6 = SIX : NA: host command packets on linetype 6 are invalid 7 = SEVEN : NA: host command packets on linetype 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

[THIS PAGE INTENTIONALLY LEFT BLANK]

25.0 HIGH-DEFINITION MULTIMEDIA INTERFACE

The High-Definition Multimedia Interface (HDMI) receives pixels from the Display Controller and audio from HDA. It combines and transmits them together.

25.1 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 SOR sequencer
- Start HDMI SOR (Serial Output Resource)
- Start display

Since audio is entirely asynchronous to HDMI or display, the SPDIF and HDA modules can be started any time.

25.1.1 Power

Turn on the necessary power supplies.

- MIPI_PLL: 1.2V
- HDMI_AVDD: 3.3V
- VHDCP: 2.8V (shared with LCD CORE, KeyPAD_LED, TouchPanel VDD)

These need to be enabled in PMU via I2C.

25.1.2 Clocks

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 13 MHz oscillator, using PLLD, would be:

Pixel Clock MHz	PLLD Freq	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)	PLLD_OUT0	PLLD_OUT0
74.25	594/2	13	594	0x6 (/4)	0x6 (/4)	PLLD_OUT0	PLLD_OUT0
148.5	594/2	13	594	0x2 (/2)	0x2 (/2)	PLLD_OUT0	PLLD_OUT0

In addition, the spdif_out and HDA2CODEC_2X modules must derive from same oscillator as display and HDMI, since audio and video clock frequencies must be a fixed ratio. The HDA2CODEC_2X clock must be enabled and configured for 54 MHz. For HDA Audio, additionally, the HDA2HDMICODEC clock must be enabled. 6 clock use cases with permutations of 4 clock enables are defined in Tegra® 3 devices.

25.1.3 Display Timing

Frame timing is controlled by display. HDMI has certain requirements, however – imposed by both the module itself, and the HDMI standards.

HDMI gets the following signals from display:

- h_blank – derived in HW from h_ref_to_sync, h_sync_width, h_back_porch, h_disp_active, h_front_porch
- v_blank – derived in HW 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 format	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

The video_preamble uses h_pulse2, with 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

Display Datapath

Dithering should be disabled, since HDMI uses full 8 bits per component.

For limited-range modes (i.e., everything except VGA 640x480), R/G/B should be scaled to lie in [16, 235] nominal range; this can be done in color palette and/or in CSC block. See the HDMI specification, section 6.6. For most HDMI modes, codes 0 and 255 must be removed; HDMI's NV_PDISP_INPUT_CONTROL.ARM_VIDEO_RANGE = LIMITED enables this.

25.1.4 HDMI Basics

- Select display source with NV_PDISP_HDMI_SRC_SELECT
- Program SOR reference clock, based on HDMI pixel clock – see NV_PDISP_SOR_REFCLK in spec. 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, power down.

25.1.5 TMDS Macro Configuration

The table below gives values for various modes. Registers not listed should use the initial (reset) values.

Register	480p	720p	1080p
SOR_LANE_DRIVE_CURRENT.LANE0..3	0xa	0xa	0xa
SOR_PLL1.TMDS_TERM	ENABLE	ENABLE	ENABLE
SOR_PLL1.TMDS_TERMADJ	0x0	0x0	0x0
SOR_PLL0.VCOCAP	0x3	0x3	0x3
SOR_PLL0.BG_V17_S	0x3	0x3	0x3
SOR_PLL1.PE_EN	ENABLE	ENABLE	ENABLE
PE_CURRENT0..3	0xa	0xa	0xa

Note: NV_PDISP_SOR_LANE_DRIVE_CURRENT.FUSE_OVERRIDE is TRUE.

25.1.6 HDA Controller Init

Basic operations:

- HDA IPFS init, FPCI, and HDA config 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.

25.1.7 HDA Audio Codec Init

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, out-of-box driver should correctly discover and configure the codec. See the HD Audio specification. A summary of 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
   a. Bit 0: 1 (DigEn=1)
   b. Bit 3..6: channel status pre/copy/audio/pro bits
   c. Bit 7: channel status CC bit 7 (called GEN_LEVEL)
4. AOC widget: SET_DIGITAL_CONV_CONTROL2:

a. 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.

25.1.8 Audio

Both “Hardware CTS” and “Force SW CTS” modes must be used for generating ACR packets, Set NV_PDISP_HDMI_ACR_CTRL (all fields) to 0

1. Reset N counter: HDMI.NV_PDISP_AUDIO_N.RESETF = ASSERT, GENERATE=ALTERNATE, LOOKUP=DISABLE
2. 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 spec section 7.2.3). LSB of N goes in SB6, and MSB into SB4.
3. Likewise, program the appropriate ACR CTS value into NV_PDISP_HDMI_ACR_0441_SUBPACK_LOW. LSB of CTS goes in SB3 and MSB into SB1.
4. Program computed AVAL into appropriate SOR_AUDIO_AVAL_ register
5. Set NV_PDISP_AUDIO_N.VALUE = N – 1 (from above)
6. Set NV_PDISP_HDMI_ACR_0441_SUBPACK_HIGH.ENABLE = YES.
7. Set NV_PDISP_HDMI_SPARE = 0x10003
8. Bring N counter of of reset: set HDMI.NV_PDISP_AUDIO_N.RESETF = DEASSERT
9. Program audio source with SOR_AUDIO_CNTRL0.SOURCE_SELECT = SPDIF for spdif, HDAL for HDA, or AUTO. AUTO automatically selects HDA if HDMI’s internal HDA codec has been enumerated by HDA controller, else SPDIF
10. 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.

25.1.9 Other HDMI Configuration

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:
   - 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 HDA Codec if desired, using one of its Generic packets. If so, don’t 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 CEA-861-D spec. 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 GENERIC InfoFrame to send HDMI Vendor Specific Infoframe. When using HDA Audio, this *may* come from HDA Codec if desired, using one of its Generic packets. If so, don't 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

25.1.10 Start HDMI SOR

1. Enable TMDS macro with SOR_PLL0.PWR=0, VCOPD=0, PULLDOWN=0.
2. write SOR_PWR with NORMAL_STATE = PU, SAFE_STATE = PD, SETTING_NEW = TRIGGER
3. write SOR_PWR with same but 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

25.1.11 Start Display

- Set DISP_WIN_OPTIONS.HDMI_ENABLE = ENABLE
- Start display the usual way with DISPLAY_COMMAND.DISPLAY_CTRL_MODE and DISPLAY_POWER_CONTROL.

25.2 HDCP

HDCP requires the presence of a correctly programmed KFUSE, as well as upstream C keys provided by software. Two main sequences are downstream authentication and upstream authentication. Downstream authentication actually permits encryption of the HDMI output, and involves a handshake between 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 downstream receiver uses DDC, which requires I2C.

25.2.1 HDCP KFUSE Key Copying

HDCP keys are stored encrypted in kfuse. Before starting HDCP, SW must copy the keys (576 bytes) from the KFUSE to HDMI registers.

1. To enable use of kfuse keys, set NV_PDISP_KEY_CTRL.LOCAL_KEYS = ENABLED.
2. Bring kfuse module out of reset and enable clock, if not already. This can be before or after HDMI is out of reset.
3. Wait for kfuse to read the keys from 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 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.

25.2.2 Downstream Authentication

This can be done before enabling transmission.

1. Set RG_HDCP_CTRL.ONEONE = ENABLE. RG_HDCP_CTRL.RUN = YES. Causes 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 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 receiver via I2C.
6. Read Aksv from RG_HDCP_AKSV_LSB/MSB, check validity (20 high bits), and write to receiver via I2C.

7. Poll for RG_HDC_CTRL.R0 = VALID
8. Read RG_HDCP_RI and compare with receiver's RI – should match.

If the above 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.

25.2.3 Upstream Authentication

The HDMI module does not handle the protocol itself. Rather, it provides primitives that SW 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's 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

SW 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's 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

25.3 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 specs.

25.3.1 Hot-plug and ELD (EDID-like data)

When HDMI driver receives notification of hot-plug interrupt, it reads downstream TV's E-EDID ROM. The EDID contains audio-related data, so the audio driver needs it too. HDMI can send the hot-plug event and EDID buffer to Audio driver via HDA Codec.

Note:	See register comments in SOR_AUDIO_HDA_PRESENSE and SOR_AUDIO_HDA_ELD_BUFWR later in this section.

25.3.2 Copy-Protection

Audio driver can query copy-protection state (i.e., HDCP enabled), and request 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.

25.3.3 InfoPacket

In HDA mode, the Audio Infoframe data is provided by internal HDA Codec, not by the HDMI_AUDIO_INFOFRAME* registers. The SET_DIP_INDEX/SET_DIP_DATA verbs write this data.

In addition, 3 additional generic infopackets are provided by HDA Codec, in addition to the exiting generic packet.

25.3.4 Miscellaneous

For Audio to HDMI signaling, HDA Codec has SCRATCH verbs that are reflected in 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 HDA Codec GET_NV_SCRATCH_REGN_FROM_DISP verb and cause an Unsolicited Response (interrupt in HDA Controller).

25.4 Clock Use Cases

Tegra 3 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

25.5 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));

}
```

25.6 HDMI Registers

25.6.1 HDMI_CTXSW_0

Channel IDs

Context switch register. Should be common to all modules. Includes the current channel/class (which is writable by SW) 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). SW 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: 000h | Read/Write: R/W | Reset: 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

Bit	R/W	Reset	Description
			0 = MANUAL 1 = AUTOACK
9:0	RW	0x0	CURR_CLASS: Current working class

25.6.2 HDMI_NV_PDISP_SOR_STATE0_0

Writing a 1 to this field cause a 1cycle pulse which is used to activate the fields in registers NV_PDISP_SOR_STATE[12]. 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 seems to be flopped to create read-only field "attached" in register NV_PDISP_SOR_TEST.

Offset: 001h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	UPDATE

25.6.3 HDMI_NV_PDISP_SOR_STATE1_0

Offset: 002h | Read/Write: R/W | Reset: 0b00000

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

25.6.4 HDMI_NV_PDISP_SOR_STATE2_0

Offset: 003h | Read/Write: R/W | Reset: 0b000000101010001

Bit	Reset	Description
14	0x0	ASY_DEPOL: 0 = POSITIVE_TRUE 1 = NEGATIVE_TRUE
13	0x0	ASY_VSYNCPOL: 0 = POSITIVE_TRUE 1 = NEGATIVE_TRUE
12	0x0	ASY_HSYNCPOL: 0 = POSITIVE_TRUE 1 = NEGATIVE_TRUE
11:8	SINGLE_TMDS_A	ASY_PROTOCOL: 1 = SINGLE_TMDS_A 15 = CUSTOM
7:6	COMPLETE_RASTER	ASY_CRCMODE: 0 = ACTIVE_RASTER 1 = COMPLETE_RASTER 2 = NON_ACTIVE_RASTER

Bit	Reset	Description
5:4	SUBHEAD0	ASY_SUBOWNER: 0 = NONE 1 = SUBHEAD0 2 = SUBHEAD1 3 = BOTH
3:0	HEAD0	ASY_OWNER: 0 = NONE 1 = HEAD0

25.6.5 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 004h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

25.6.6 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 005h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

25.6.7 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 006h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	VALUE: 0 = ZERO

25.6.8 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 007h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	VALUE: 0 = ZERO

25.6.9 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 008h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	VALUE: 0 = ZERO

25.6.10 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 009h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

25.6.11 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 00ah | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxxx00000000

Bit	Reset	Description
31	0x0	REPEATER: 0 = VALUE
7:0	0x0	VALUE: 0 = ZERO

25.6.12 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 00bh | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	VALUE: 0 = ZERO

25.6.13 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, 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: 00ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	VALUE: 0 = ZERO

25.6.14 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 00dh | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	VALUE: 0 = ZERO

25.6.15 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 00eh | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	VALUE: 0 = ZERO

25.6.16 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 00fh | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

25.6.17 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 driver to start or stop the downstream protocol
- CRYPT: Set by driver to turn encryption on.
- Write ENABLED to enable, but only after RUN == YES. Once enabled, is disabled by writing RUN=NO.
- DUAL_LINK_EN: Set by 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, 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 spec for EESS signaling. See Chapter 2.2 in the HDCP 1.1 spec 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: 010h | Read/Write: R/W | Reset: 0bxxxxxxxxxx0000

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

25.6.18 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, 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: 011h | Read/Write: R/W | Reset: 0b00000001

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

Bit	Reset	Description
		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

25.6.19 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 012h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

25.6.20 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, the Upstream Link for HDCP specification document for more information.

Usage: normal operation

Offset: 013h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

25.6.21 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, 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, none of our chips implement "Read Z".

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 isn't 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: 014h | Read/Write: RO | Reset: 0bxxxxxxxx

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
4	X	STATUS_INTPNL: 0 = INACTV 1 = ACTV
3:0	X	STATUS_MAX_CMODE_IDX

25.6.22 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, 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 isn't being encrypted. Data being transmitted by a TMDS link is also considered unprotected if it isn't being encrypted,*except* for the case where

the link feeds the internal panel of a laptop. The HDCP spec 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 isn't 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: 015h | Read/Write: RO | Reset: 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

25.6.23 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 016h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	VALUE: 0 = ZERO

25.6.24 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 017h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	VALUE: 0 = ZERO

25.6.25 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, the Upstream Link for HDCP specification for more information.

Usage: normal operation

Offset: 018h | Read/Write: RO | Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	RESERVED

25.6.26 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: 019h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	RESERVED

25.6.27 HDMI_NV_PDISP_HDMI_AUDIO_INFOFRAME_CTRL_0

Please refer IEC60958-1 for general and IEC60958-3 for consumer spec, for related audio information.

Additional notes regarding the ACR packet:

There are several methods that can used to generate the N and CTS values in the Audio Clock Regeneration packet. The details of these methods are described here.

Full SW control: SW controls the contents of the possible ACR packets. SW writes the contents of the ACR_XXXX_SUBPACK_HIGH and ACR_XXXX_SUBPACK_LOW for all seven audio frequencies.

SW 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 SW (FREQS)

SW 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: SW 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.

SW writes _ENABLE to HDMI_SPARE_HW_CTS to enable this feature.

SW 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.

SW sets all ENABLEs in ACR_CTRL to _NO. (The easy way to do this is just to write zero to the whole register.)

SW 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. (SW does not need to write N to these places if Hardware Selected N is used.)

SW 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.)

SW writes _USE_HW_CTS_VAL to ACR_0441_SUBPACK_LOW_SB1.

Hardware selected N: SW 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.

SW writes the 7 possible N values into the 7 AUDIO_NVAL registers. This is based on pixel clock frequency.

SW enables this feature by setting AUDIO_N_LOOKUP to _ENABLE. HW will used 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. HW 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 spec 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 spec.

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, 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 spec for more information

CHKSUM_HW: HW provides a way to calculate the Checksum for the infoframes.

_ENABLE will enable the HW calculation to be passed to the packet

_DISABLE :

1. Will use the register value defined in NV_PDISP_SOR_HDMI_AUDIO_INFOFRAME_SUBPACK0_LOW_PB0 for the SPDIF

2. Will use the Check Sum provided by the Azalia codec for the Azalia audio formats.

Usage: normal operation

Offset: 01eh | Read/Write: R/W | Reset: 0b10xxx0xxx0

Bit	Reset	Description
9	ENABLE	CHKSUM_HW: 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

25.6.28 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: 01fh | Read/Write: RO | Reset: 0bx

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

25.6.29 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 spec. HB0, HB1, and HB2 are defined fields of the header from the spec.

Offset: 020h | Read/Write: R/W | Reset: 0b000000000000000000000000

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

25.6.30 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 spec. See chapter 8.2.2 - Audio InfoFrame, Table 8-5, in the HDMI spec for more information

Offset: 021h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

25.6.31 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 spec. See chapter 8.2.2 - Audio InfoFrame, Table 8-5, in the HDMI spec for more information

Offset: 022h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:8	0x0	PB5
7:0	0x0	PB4

25.6.32 HDMI_NV_PDISP_HDMI_AVI_INFOFRAME_CTRL_0

The following registers are used to send a fourteen-byte packet intended for sending avi infoframe packet as described in section 8.2.1 of the HDMI spec.

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 spec for more information

Offset: 023h | Read/Write: R/W | Reset: 0b10xxx0xxx0

Bit	Reset	Description
9	ENABLE	CHKSUM_HW: 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

25.6.33 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: 024h | Read/Write: RO | Reset: 0bx

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

25.6.34 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 spec. HB0, HB1, and HB2 are defined fields of the header from the spec.

Offset: 025h | Read/Write: R/W | Reset: 0b000000000000000000000000

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

25.6.35 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 spec. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI spec for more information.

Offset: 026h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

25.6.36 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 spec. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI spec for more information.

Offset: 027h | Read/Write: R/W | Reset: 0b000000000000000000000000

Bit	Reset	Description
23:16	0x0	PB6
15:8	0x0	PB5
7:0	0x0	PB4

25.6.37 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 spec. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI spec for more information.

Offset: 028h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	PB10
23:16	0x0	PB9
15:8	0x0	PB8
7:0	0x0	PB7

25.6.38 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 spec. See chapter 8.2.1 - Auxiliary Video information (AVI) InfoFrame, Table 8-2, in the HDMI spec for more information.

Offset: 029h | Read/Write: R/W | Reset: 0b000000000000000000000000

Bit	Reset	Description
23:16	0x0	PB13
15:8	0x0	PB12
7:0	0x0	PB11

25.6.39 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: 02ah | Read/Write: R/W | Reset: 0b1xxx0xxx0xxx0xxx0

Bit	Reset	Description
16	EN	AUDIO: 0 = DIS 1 = EN
12	DIS	HBLANK: 0 = DIS 1 = EN
8	DIS	SINGLE: 0 = DIS

Bit	Reset	Description
		1 = EN
4	DIS	OTHER: 0 = DIS 1 = EN
0	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

25.6.40 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: 02bh | Read/Write: RO | Reset: 0bx

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

25.6.41 HDMI_NV_PDISP_HDMI_GENERIC_HEADER_0

HDMI_GENERIC_HEADER

This register should be written with the contents of the packet header.

Offset: 02ch | Read/Write: R/W | Reset: 0b000000000000000000000000

Bit	Reset	Description
23:16	0x0	HB2
15:8	0x0	HB1
7:0	0x0	HB0

25.6.42 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK0_LOW_0

HDMI_GENERIC_SUBPACK0_LOW

Bytes 0-3 of the packet are written into this register.

Offset: 02dh | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	PB3
23:16	0x0	PB2
15:8	0x0	PB1
7:0	0x0	PB0

25.6.43 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK0_HIGH_0

HDMI_GENERIC_SUBPACK0_HIGH

Bytes 4-6 of the packet are written into this register.

Offset: 02eh | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
23:16	0x0	PB6
15:8	0x0	PB5
7:0	0x0	PB4

25.6.44 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK1_LOW_0

HDMI_GENERIC_SUBPACK1_LOW

Bytes 7-10 of the packet are written into this register.

Offset: 02fh | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	PB10
23:16	0x0	PB9
15:8	0x0	PB8
7:0	0x0	PB7

25.6.45 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK1_HIGH_0

HDMI_GENERIC_SUBPACK1_HIGH

Bytes 11-13 of the packet are written into this register.

Offset: 030h | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
23:16	0x0	PB13
15:8	0x0	PB12
7:0	0x0	PB11

25.6.46 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK2_LOW_0

HDMI_GENERIC_SUBPACK2_LOW

Bytes 14-17 of the packet are written into this register.

Offset: 031h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	PB17
23:16	0x0	PB16
15:8	0x0	PB15

Bit	Reset	Description
7:0	0x0	PB14

25.6.47 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK2_HIGH_0

HDMI_GENERIC_SUBPACK2_HIGH

Bytes 18-20 of the packet are written into this register.

Offset: 032h | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
23:16	0x0	PB20
15:8	0x0	PB19
7:0	0x0	PB18

25.6.48 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK3_LOW_0

HDMI_GENERIC_SUBPACK3_LOW

Bytes 21-24 of the packet are written into this register.

Offset: 033h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	PB24
23:16	0x0	PB23
15:8	0x0	PB22
7:0	0x0	PB21

25.6.49 HDMI_NV_PDISP_HDMI_GENERIC_SUBPACK3_HIGH_0

HDMI_GENERIC_SUBPACK3_HIGH

Bytes 25-27 of the packet are written into this register.

Offset: 034h | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
23:16	0x0	PB27
15:8	0x0	PB26
7:0	0x0	PB25

25.6.50 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 spec.

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: 035h | Read/Write: R/W | Reset: 0b0010xxxxxxx1xxxxxxx0xxxxxxx0

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

25.6.51 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 spec for more information.

Offset: 036h | Read/Write: R/W | Reset: 0b000000000000000000000000

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

25.6.52 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 spec 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: 037h | Read/Write: R/W | Reset: 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

25.6.53 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 spec for more information.

Offset: 038h | Read/Write: R/W | Reset: 0b00000000000000000000000

Bit	Reset	Description
31:24	0x0	SB1: 0 = USE_HW_CTS_VAL
23:16	0x0	SB2
15:8	0x0	SB3

25.6.54 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 spec 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 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: 039h l Read/Write: R/W l Reset: 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

25.6.55 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 spec for more information.

Offset: 03ah l Read/Write: R/W l Reset: 0b000000000000000000000000

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

25.6.56 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 spec 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: 03bh l Read/Write: R/W l Reset: 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

25.6.57 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 spec for more information.

Offset: 03ch l Read/Write: R/W l Reset: 0b000000000000000000000000

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

25.6.58 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 spec 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: 03dh l Read/Write: R/W l Reset: 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

25.6.59 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 spec for more information.

Offset: 03eh l Read/Write: R/W l Reset: 0b000000000000000000000000

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

25.6.60 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 spec 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: 03fh l Read/Write: R/W l Reset: 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

25.6.61 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 in the HDMI spec for more information.

Offset: 040h l Read/Write: R/W l Reset: 0b000000000000000000000000

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

25.6.62 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 spec 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: 041h l Read/Write: R/W l Reset: 0b0xxxxxxx00000000000000000000000

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
23:16	0x0	SB4

Bit	Reset	Description
15:8	0x0	SB5
7:0	0x0	SB6

25.6.63 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 spec for more information.

Offset: 042h l Read/Write: R/W l Reset: 0b000000000000000000000000

Bit	Reset	Description
31:24	0x0	SB1
23:16	0x0	SB2
15:8	0x0	SB3

25.6.64 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 spec 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: 043h l Read/Write: R/W l Reset: 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

25.6.65 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 spec. 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 HW (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 spec, 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 HW based or SW based. CT field is required in the construction of Audio Infoframe packet (HDMI Spec sec 8.2.2). By default CT field is constructed by SW but by setting this bit CT field in the construction of the Audio Infoframe packet comes as a sideband signal from the hdacodec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field has no meaning

CC_SELECT: Diagnostic workaround bit to decide whether the value of "channel count" will be HW based or SW based. CC field is required in the construction of Audio Infoframe packet (HDMI Spec sec 8.2.2). By default CC field is constructed by SW but by setting this bit CC field in the construction of the Audio Infoframe packet comes as a sideband signal from the hdacodec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field has no meaning

SF_SELECT: Diagnostic workaround bit to decide whether the value of "sampling frequency" will be HW based or SW based. SF field is required in the construction of Audio Infoframe packet (HDMI Spec sec 8.2.2). By default SF field is constructed by SW but by setting this bit SF field in the construction of the Audio Infoframe packet comes as a sideband signal from the hdacodec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field has no meaning

SS_SELECT: Diagnostic workaround bit to decide whether the value of "sample size" will be HW based or SW based. SS field is required in the construction of Audio Infoframe packet (HDMI Spec sec 8.2.2). By default SS field is constructed by SW but by setting this bit SS field in the construction of the Audio Infoframe packet comes as a sideband signal from the hdacodec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field has no meaning

CA_SELECT: Diagnostic workaround bit to decide whether the value of "channel allocation" will be HW based or SW based. CA field is required in the construction of Audio Infoframe packet (HDMI Spec sec 8.2.2). By default CA field is constructed by SW but by setting this bit CA field in the construction of the Audio Infoframe packet comes as a sideband signal from the hdacodec. If NV_CHIP_DISP_EXTENDED_AUD_FMT is not defined in projects.spec, this field has no meaning

ENABLE: Set to _YES to enable HDMI for this head. Set to _NO to disable HDMI for this head.

Offset: 044h l Read/Write: R/W l Reset: 0b0x00000xxx00010xxx0x0x0x0111000

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

Bit	Reset	Description
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

25.6.66 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 spec.

END: Defines the end of the keepout period. This is measured in pixels after the active edge of VSYNC. This is defined in the spec and should not need to be written beyond its init 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 spec and should not need to be written beyond its init 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 should be _YES. When set to _NO, the keepout window is ignored and HDMI packets may be sent regardless of the keepout window.

Offset: 045h l Read/Write: R/W l Reset: 0b1xxxxx0110011010xxxxxx1010001010

Bit	Reset	Description
31	YES	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
25:16	0x19a	START
9:0	0x28a	END

25.6.67 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 spec.

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 spec and should not need to be written beyond its init 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 spec and should not need to be written beyond its init value.

ENABLE: _YES will allow EESS signaling during the window of opportunity. _NO will prevent EESS signaling.

Offset: 046h l Read/Write: R/W l Reset: 0b1xxxxx1000000000xxxxxx1000010000

Bit	Reset	Description
31	YES	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
25:16	0x200	START
9:0	0x210	END

25.6.68 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 spec. 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: 047h l Read/Write: R/W l Reset: 0b0xxx0xxx0

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

25.6.69 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: 048h l Read/Write: RO l Reset: 0bx

Bit	Reset	Description
0	X	SENT: 1 = DONE 0 = WAITING

25.6.70 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 HMDI spec for more information.

Offset: 049h l Read/Write: R/W l Reset: 0b00000000000000000000000001

Bit	Reset	Description
23:16	0x0	SB2
15:8	0x0	SB1
7:0	0x1	SB0: 1 = SET_AVMUTE 16 = CLR_AVMUTE

25.6.71 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 spec 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 spec.

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 spec 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: 04ah l Read/Write: R/W l Reset: 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

25.6.72 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_LENTGH=_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: 04bh l Read/Write: R/W l Reset: 0b0xxxxxxxxxxxxxxxxxxxxxxx11110001

Bit	Reset	Description
31	NO	ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN
7:4	0xf	ORIGINAL:

Bit	Reset	Description
		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

25.6.73 HDMI_NV_PDISP_HDMI_EMU0_0

HDMI_EMU0 and 1 control hdmi_decoder bfm. They do indirect addressing. EMU0 is address, and EMU1 is 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 propogate

3. read data register

Offset: 04ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	regx

25.6.74 HDMI_NV_PDISP_HDMI_EMU1_0

Offset: 04dh | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	regx

25.6.75 HDMI_NV_PDISP_HDMI_EMU1_RDATA_0

Offset: 04eh | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	0x0	rdata

25.6.76 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 spec. In most cases the value in the "Other" row can be used.

FORCE_SW_CTS: When HW_CTS is ENABLE, it uses measured CTS. However, when the audio clock / video clock ratio is known, this is undesirable. Currently, HW_CTS can't 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 SW 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

SW 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: 04fh l Read/Write: R/W Reset: 0b00000000000000010000000000000000

Bit	Reset	Description
31:0	0x10000	Reserved

25.6.77 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 SPDIF 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 (ie. compressed audio)

COPYRIGHT Copyright status of the audio.

- -YES: Copyright is asserted
- -NO: Copyright is not asserted

D: The value 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 50microseconds/15microseconds 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 spec 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: 050h l Read/Write: RO l Reset: 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

25.6.78 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_LENTGH=_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: 051h l Read/Write: RO l Reset: 0bxxxxxxxx

Bit	Reset	Description
7:4	X	ORIGINAL: 0 = UNDEFINED
3:1	X	LENGTH: 0 = MAX20_UNDEF

Bit	Reset	Description
		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

25.7 HDCPRIF REGISTERS

This is the register space for the HDCP ROM interface control registers. These register controls writing the contents of the HDCPRIF_KEY registers (56 bits) into the local key store. The keys are decoded by the CD, which outputs them on the internal key data bus. The key registers are then written by overloading the priv reg bus with the key data when a write to either key register is executed. The key data must then be input into the key store using the control commands below. As no access to key data from public busses is allowed, all these operations are performed as hardware controlled functions that are initiated by software.

Typical operation is as follows:

1. Write to the CTRL register with LOCAL_ENABLED, AUTOINC_DISABLED, WRITE5_INIT, WRITE7_INIT, WRITE1_INIT, LOAD_SELECT_ZERO, LOAD_ADDRESS_ZERO.
2. Decode the downstream KSV key data in the CD block.
3. Write NV_PDISP_HDCPRIF_KEY(0) and NV_PDISP_HDCPRIF_KEY(1) with zeroes. This operation will be overloaded with the key data from the CD block.
4. Write to the CTRL register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE5_TRIGGER, WRITE7_INIT, WRITE1_INIT, LOAD_SELECT_ZERO, LOAD_ADDRESS_ZERO. Poll for WRITE5_DONE.
5. Decode the downstream key in the CD block.
6. Write NV_PDISP_HDCPRIF_KEY(0) and NV_PDISP_HDCPRIF_KEY(1) with zeroes. This operation will be overloaded with the key data from the CD block.
7. Write to the CTRL register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE7_TRIGGER, WRITE5_INIT, WRITE1_INIT, LOAD_SELECT_ZERO, LOAD_ADDRESS_ZERO. Poll for WRITE7_DONE.
8. Repeat 5 through 7 39 more times. There are always 40 keys and one KSV value.

If upstream key authentication is required, do the following:

9. Decode the upstream KSV key data in the CD block.
10. Write NV_PDISP_HDCPRIF_KEY(0) and NV_PDISP_HDCPRIF_KEY(1) with zeroes. This operation will be overloaded with the key data from the CD block.
11. Write to the CTRL register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE5_TRIGGER, WRITE7_INIT, WRITE1_INIT, LOAD_SELECT_ZERO, LOAD_ADDRESS_ZERO. Poll for WRITE5_DONE.
12. Decode the first upstream key in the CD block.
13. Write NV_PDISP_HDCPRIF_KEY(0) and NV_PDISP_HDCPRIF_KEY(1) with zeroes. This operation will be overloaded with the key data from the CD block.
14. Write to the CTRL register with LOCAL_ENABLED, AUTOINC_ENABLED, WRITE7_TRIGGER, WRITE5_INIT, WRITE1_INIT, LOAD_SELECT_ZERO, LOAD_ADDRESS_ZERO. Poll for WRITE7_DONE.

15. Repeat 12 through 14 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: 1'st downstream key

Byte 12 to Byte 18: 2'nd downstream key

...

Byte 278 to Byte 284: 40'th downstream key

Byte 285 to Byte 289: upstream KSV

Byte 290 to Byte 296: 1'st upstream key

...

Byte 563 to Byte 569: 40'th upstream key

25.7.1 HDMI_NV_PDISP_SOR_CAP_0

SOR (Serial Output Resource): The registers are defined as if all the sors are fully featured (e.g. dual link). If a feature is not supported, the register remains accessible, i.e. writes will not hang, reads will always return the default value.

Each SOR unit consists of two sub-links designated A and B. In single link operation, the A and/or B sub-link is active. In dual link modes, both links are active. SOR units consist of either one or two links operating in the following combinations:

- Sub-link A (single link mode)
- Sub-link B (single link mode)
- Sub-link AB (dual single link copy mode)
- Sub-links A+B (dual link mode)
- Sub-links A,B (dual single link mode) planned for future enhanced zaphod mode.

Driver level control of the SOR is accomplished through methods in the core channel of the display engine. However, certain aspects of SOR operation are chip, process, voltage, and pixel clock dependent. These need to be controlled via registers by the VBIOS or RM during mode configuration. In addition, the preset LVDS mode may occasionally require some tweaking (to handle certain odd panels), so registers are provided to customize the support of LVDS.

This register reports the innate capabilities of the sor. Note that depending on the actual board build configuration, these values may not be exactly the same ones reflected in the overall capabilities structure reported through the class. With the exception of the last bit, the values assumed by these fields are automatically produced by the hw. The last bit represents the state of a fuse per sor that can be blown to constrain the sor to be usable only for LVDS.

This assists the determination of security for HDCP encrypted output operation. If the fuse is blown for lvds only, then the sor module hardware will enforce this. In addition, the fuse value must be taken into account when reporting overall capabilities.

Note: THIS REGISTER DEFINITION MUST BE KEPT COHERENT WITH DSI_SOR_CAP(i) !

Usage: boot / initialization

Offset: 054h l Read/Write: RO l Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31	X	LVDS_ONLY: 0 = FALSE 1 = TRUE

Bit	Reset	Description
20	X	DDI: 0 = FALSE 1 = TRUE
16	X	SDI: 0 = FALSE 1 = TRUE
11	X	DUAL_TMDS: 0 = FALSE 1 = TRUE
10	X	DUAL_SINGLE_TMDS: 0 = FALSE 1 = TRUE
9	X	SINGLE_TMDS_B: 0 = FALSE 1 = TRUE
8	X	SINGLE_TMDS_A: 0 = FALSE 1 = TRUE
3	X	DUAL_LVDS_24: 0 = FALSE 1 = TRUE
2	X	DUAL_LVDS_18: 0 = FALSE 1 = TRUE
1	X	SINGLE_LVDS_24: 0 = FALSE 1 = TRUE
0	X	SINGLE_LVDS_18: 0 = FALSE 1 = TRUE

25.7.2 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 HW and SW to deal with the sor's. At boot, the SW 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, SW should load in the new value and write SETTING_NEW=TRIGGER. Once the change has been successfully completed, the SETTING_NEW will indicate DONE. This is the state that SW 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, SW 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 HW will use whenever it initiates the mode switch/shutdown procedure for this pior. 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 i.e. 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, SW 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 doesn't happen within one frame time, then SW 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: 055h l Read/Write: R/W l Reset: 0b0xxxxxxxxxxxxx00xxxxxxxxxxxxxx00

Bit	R/W	Reset	Description
31	RW	0x0	SETTING_NEW: 0 = DONE 1 = PENDING 1 = TRIGGER
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: 0 = PD 1 = PU

25.7.3 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 one, walking zero, all one, all zero.

ACT_HEAD_OPMODE: Report the current OR operating mode.

ATTACHED: Report whether the OR is currently attached to a head.

DSRC: When DSRC=NORMAL, then 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 hw 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 hi-speed serializer logic, or post the hi-speed seralizer/deserializer logic.

TESTMUX[7:0]: Test MUX select - output seen on PROBE

Usage: debug

Offset: 056h l Read/Write: R/W l Reset: 0b000000001000xx00xxxxxxxxx0xxxx0

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

Bit	R/W	Reset	Description
			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

25.7.4 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 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 = powerdown the TMDS pll

PDBG: 1 = powerdown the bandgap

VCOPD: 1 = powerdown the VCO

PDPORT: 1 = powerdown the output drivers

RESISTORSEL: Selects the internal resistor (0) or external resistor (1).

PULLDOWN: Weak pulldown 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 reg field does not read the final pulldown value. In other words if sw writes NV_PDISP_SOR_PLL0_PULLDOWN to DISABLED and sequencer pulldown is enabled, then 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]: Field selects the loop filter and adjusts the filter resistor value.

ICHPMP[3:0]: Specifies additions to the charge pump current in steps of 0.375uA.

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 grn 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: 057h l Read/Write: R/W l Reset: 0b000010xxxx000000110011xx111111

Bit	Reset	Description
29:28	0x0	TX_REG_LOAD: 0 = RESETV
27:24	0x2	ICHPMP: 2 = RESETV
19:16	0x0	FILTER: 0 = RESETV
15:12	0x3	BG_V17_S: 3 = RESETV
11:8	0x3	VCOCAP: 3 = RESETV
5	0x1	PULLDOWN: 0 = DISABLE 1 = ENABLE
4	0x1	RESISTORSEL: 0 = INT 1 = EXT 1 = RESETV

Bit	Reset	Description
3	0x1	PDPORT: 0 = ON 1 = OFF
2	0x1	VCOPD: 0 = RESCIND 1 = ASSERT
1	0x1	PDBG: 0 = ON 1 = OFF
0	0x1	PWR: 0 = ON 1 = OFF

25.7.5 HDMI_NV_PDISP_SOR_PLL1_0

Usage: boot / initialization / mode switch

Offset: 058h l Read/Write: R/W l Reset: 0b000xxxx0000xxxxxxx00000

Bit	Reset	Description
30	SINGLE	S_D_PIN_PE: 0 = SINGLE 1 = DIFFERENTIAL
29	HALF	HALF_FULL_PE: 0 = HALF 1 = FULL
28	DISABLE	PE_EN: 0 = DISABLE 1 = ENABLE
23:20	0x0	LOADADJ: 0 = CENTER
12:9	0x0	TMDS_TERMADJ
8	0x0	TMDS_TERM: 0 = DISABLE 1 = ENABLE

25.7.6 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: 059h l Read/Write: R/W l Reset: 0b00000000

Bit	Reset	Description
23	0x0	AUX7
22	0x0	AUX6
21	0x0	AUX5

Bit	Reset	Description
20	0x0	AUX4
19	0x0	AUX3
18	0x0	AUX2
17	0x0	AUX1
16	0x0	AUX0

25.7.7 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 SW 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 powerdown the pin.
- PD_TXDB[3:0]: Bitwise control to power down the data pins of link B. Set to DISABLE to powerdown 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. 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 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 10x pixel clock so ROTVAL should be between 0-9. For LVDS, sclk is 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 rotate 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 HW 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
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 diagnostic workaround register set. SW should never need to use it. Also, note that if LVDS_ONLY=TRUE, then this register cannot be used as the HW 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: 05ah l Read/Write: R/W l Reset: 0b0000000xx0x000111001x0000000000

Bit	Reset	Description
30:28	0x0	ROTDAT: 0 = RESETV
27:24	0x0	ROTCLK: 0 = RESETV
21	0x0	PLLDIV: 0 = BY_7 1 = BY_10
19	0x0	BALANCED:

Bit	Reset	Description
		0 = DISABLE 1 = ENABLE
18	0x0	NEW_MODE: 0 = DISABLE 1 = ENABLE
17	0x0	DUP_SYNC: 0 = DISABLE 1 = ENABLE
16	0x1	LVDS_EN: 0 = DISABLE 1 = ENABLE
15	0x1	LINKACTB: 0 = DISABLE 1 = ENABLE
14	0x1	LINKACTA: 0 = DISABLE 1 = ENABLE
13:12	0x0	MODE: 0 = LVDS 1 = TMDS
11	0x1	UPPER: 1 = TRUE 0 = FALSE
9	0x0	PD_TXCB: 1 = DISABLE 0 = ENABLE
8	0x0	PD_TXCA: 1 = DISABLE 0 = ENABLE
7	0x0	PD_TXDB_3: 1 = DISABLE 0 = ENABLE
6	0x0	PD_TXDB_2: 1 = DISABLE 0 = ENABLE
5	0x0	PD_TXDB_1: 1 = DISABLE 0 = ENABLE
4	0x0	PD_TXDB_0: 1 = DISABLE 0 = ENABLE
3	0x0	PD_TXDA_3: 1 = DISABLE 0 = ENABLE
2	0x0	PD_TXDA_2: 1 = DISABLE 0 = ENABLE
1	0x0	PD_TXDA_1: 1 = DISABLE 0 = ENABLE
0	0x0	PD_TXDA_0: 1 = DISABLE 0 = ENABLE

25.7.8 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: 05bh l Read/Write: R/W l Reset: 0b0000000xxxx000x1xxxxx0x00000xxx

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
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

Bit	R/W	Reset	Description
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

25.7.9 HDMI_NV_PDISP_SOR_CRCA_0

The following registers are used to fetch crc's when running VGA mode tests. Three registers are used. Once contains the valid bit, one contains the actual computed crc, 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: verif

Offset: 05ch l Read/Write: R/W l Reset: 0b0

Bit	Reset	Description
0	0x0	VALID: 0 = FALSE 1 = TRUE 1 = RESETV

25.7.10 HDMI_NV_PDISP_SOR_CRCB_0

Usage: verif

Offset: 05dh l Read/Write: RO l Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	CRC

25.7.11 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: 05eh l Read/Write: R/W l Reset: 0bx00

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

25.7.12 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 uS timer that is used to advance the internal counters. Thus the actual time delay for the first event can be almost one uS less than what is requested. Subsequent instructions in a chain of events do transition on 1 uS boundaries, however. If a minimum spec of "x" uS is required, then it is best to program "x+1" uS of delay.

Usage: boot / initialization / mode switch

Offset: 05fh l Read/Write: R/W l Reset: 0b0xxxxxxxxxxxxxx100010000000xxxx

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

25.7.13 HDMI_NV_PDISP_SOR_SEQ_INST0_0

Usage: boot / initialization

Offset: 060h l Read/Write: R/W l Reset: 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

25.7.14 HDMI_NV_PDISP_SOR_SEQ_INST1_0

Usage: boot / initialization

Offset: 061h l Read/Write: R/W l Reset: 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

25.7.15 HDMI_NV_PDISP_SOR_SEQ_INST2_0

Usage: boot / initialization

Offset: 062h l Read/Write: R/W l Reset: 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

25.7.16 HDMI_NV_PDISP_SOR_SEQ_INST3_0

Usage: boot / initialization

Offset: 063h l Read/Write: R/W l Reset: 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

25.7.17 HDMI_NV_PDISP_SOR_SEQ_INST4_0

Usage: boot / initialization

Offset: 064h l Read/Write: R/W l Reset: 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

25.7.18 HDMI_NV_PDISP_SOR_SEQ_INST5_0

Usage: boot / initialization

Offset: 065h l Read/Write: R/W l Reset: 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

25.7.19 HDMI_NV_PDISP_SOR_SEQ_INST6_0

Usage: boot / initialization

Offset: 066h l Read/Write: R/W l Reset: 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

25.7.20 HDMI_NV_PDISP_SOR_SEQ_INST7_0

Usage: boot / initialization

Offset: 067h l Read/Write: R/W l Reset: 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

25.7.21 HDMI_NV_PDISP_SOR_SEQ_INST8_0

Usage: boot / initialization

Offset: 068h l Read/Write: R/W l Reset: 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

25.7.22 HDMI_NV_PDISP_SOR_SEQ_INST9_0

Usage: boot / initialization

Offset: 069h l Read/Write: R/W l Reset: 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

25.7.23 HDMI_NV_PDISP_SOR_SEQ_INSTA_0

Usage: boot / initialization

Offset: 06ah l Read/Write: R/W l Reset: 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

25.7.24 HDMI_NV_PDISP_SOR_SEQ_INSTB_0

Usage: boot / initialization

Offset: 06bh l Read/Write: R/W l Reset: 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

25.7.25 HDMI_NV_PDISP_SOR_SEQ_INSTC_0

Usage: boot / initialization

Offset: 06ch l Read/Write: R/W l Reset: 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

25.7.26 HDMI_NV_PDISP_SOR_SEQ_INSTD_0

Usage: boot / initialization

Offset: 06dh l Read/Write: R/W l Reset: 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

25.7.27 HDMI_NV_PDISP_SOR_SEQ_INSTE_0

Usage: boot / initialization

Offset: 06eh l Read/Write: R/W l Reset: 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

25.7.28 HDMI_NV_PDISP_SOR_SEQ_INSTF_0

Usage: boot / initialization

Offset: 06fh l Read/Write: R/W l Reset: 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

25.8 Test and Debug Registers

25.8.1 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 which has been re-parallelized from the serial data going out.

These encoded bits of data are broken up into 3 16-bits chunks 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 chunks, a CRC16 is computed, CRCH for the upper 16-bit chunk, CRCM form the middle 16-bit chunk, and CRCL for the lower 16-bit chunk. 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 previous running crc16 values are captured into the vcrc registers and a previous crc value of all zeroes are 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 aren't latched at vsync but are free-running.

The VCRC registers for sub-link A

Usage: debug

Offset: 072h l Read/Write: RO l Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	CRCM
15:0	X	CRCL

25.8.2 HDMI_NV_PDISP_SOR_VCRCA1_0

Usage: debug

Offset: 073h l Read/Write: RO l Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	CRCH

25.8.3 HDMI_NV_PDISP_SOR_CCRCA0_0

The CCRC registers contain the CRC value for the encoded link which captured when the trigger event occurred.

The CCRC registers for sub-link A

Usage: debug

Offset: 074h l Read/Write: RO l Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	CRCM
15:0	X	CRCL

25.8.4 HDMI_NV_PDISP_SOR_CCRCA1_0

Usage: debug

Offset: 075h l Read/Write: RO l Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	CRCH

25.8.5 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 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: 076h l Read/Write: RO l Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	VAL

25.8.6 HDMI_NV_PDISP_SOR_EDATAA1_0

Usage: debug

Offset: 077h l Read/Write: RO l Reset: 0bxxxxxxxx

Bit	Reset	Description
7:0	X	VAL

25.8.7 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: 078h l Read/Write: RO l Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	TX1
15:0	X	TX0

25.8.8 HDMI_NV_PDISP_SOR_COUNTA1_0

Usage: debug

Offset: 079h l Read/Write: RO l Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:0	X	TX2

25.8.9 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 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: 07ah l Read/Write: R/W l Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	VAL: 0 = RESETV

25.8.10 HDMI_NV_PDISP_SOR_DEBUGA1_0

Usage: debug

Offset: 07bh l Read/Write: R/W l Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	VAL: 0 = RESETV

25.8.11 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 larger than the number of cycles between consecutive VSYNC's, then the trigger is not reset while it is still pending to capture data. Once the data have 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: 07ch l Read/Write: R/W l Reset: 0b000000000000000000000000

Bit	Reset	Description
23:0	0x0	VAL: 0 = RESETV

25.8.12 HDMI_NV_PDISP_SOR_MSCHECK_0

The mode switch monitor is used for hw 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 are 4 bits, so the count will clamp at 15.

Usage: debug

Offset: 07dh l Read/Write: R/W l Reset: 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

25.8.13 HDMI_NV_PDISP_SOR_LANE_DRIVE_CURRENT_0

TMDS per-lane I/O current control.

Although each control is specified as 8b below, making it easy to set the value in hex, really only the 6 lsbs of each lane are used.

000000 -> 1.500 mA

000001 -> 1.875 mA

...

(each increment is an additional 0.375 mA)

...

111110 -> 24.750 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: 07eh l Read/Write: R/W l Reset: 0b0x100000xx100000xx100000xx100000

Bit	Reset	Description
31	FALSE	FUSE_OVERRIDE: 0 = FALSE 1 = TRUE
29:24	0x20	LANE3
21:16	0x20	LANE2
13:8	0x20	LANE1
5:0	0x20	LANE0

25.9 Audio Registers

25.9.1 HDMI_NV_PDISP_AUDIO_DEBUG0_0

This is the register space for the SPDIF audio decoding block. Please refer IEC60958-1 for general and IEC60958-3 for consumer specification.

AUDIO_DEBUG0

This register is meant for debugging. It indicates various possible errors in the incoming SPDIF 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, i.e., to clear the _ERROR bits, just write _RESET to the bit you want to clear.

THRESHOLD_ERROR: When THRESHOLD_LOW is bigger than THERSHOLD_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 of HDMI unit does not drain away 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 an non-192 frames' block is detected this bit will be set. Again, this indicates a problem with the incoming SPDIF stream.

BITS_ERROR: A frame should have 56 bits audio sample + 8 bits preamble. If a non 56 bits audio sample is detected in a frame, this bit will be set.

OVERFLOW_ERROR: If the spdif input stream is floating, the 10 bits pulse width counter will be overflowed, then this bit will be set. See OVERFLOW_TH for detail.

PREAMBLE_ERROR: A frame should have 56 bits audio sample + 8 bits preamble. If there is no valid preamble for a given sample, this bit will be set.

Offset: 07fh l Read/Write: R/W l Reset: 0b0xxx0xxx0xxx0xxx0xxx0xxx0xxx0

Bit	Reset	Description
24	0x0	PREAMBLE_ERROR: 0 = NO 1 = YES 1 = RESETV
20	0x0	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
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

25.9.2 HDMI_NV_PDISP_AUDIO_DEBUG1_0

AUDIO_DEBUG1

OVERFLOW_TH (bit 9:0): There is a 10 bits counter running on dispclk_audio_fs and counting spdif's input pulse length. The maximum pulse length count is 305, when dispclk=416MHz and fs=32KHz. If someone unpluged the spdif 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 spdif input is being unpluged or floating. If the 10 bits counter is larger or equal to OVERFLOW_TH, then OVERFLOW_ERROR bit will be set.

SW should not need to program this register to anything but its init value. The init 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, we will drop the packet and flag an error condition NV_PDISP_AUDIO_DEBUG0_PREAMBLE_ERROR = _YES. This register is included as a debug to allow disabling 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_CYA: 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. SW should not need to change this bit.

Offset: 080h l Read/Write: R/W l Reset: 0b1xxxxxxxxxxxxxx1xxxxxx1111111111

Bit	Reset	Description
31	YES	PREAMBLE_RECOVER_CYA: 0 = NO 1 = YES
16	0x1	PACKET_NOISE_FILTER_ENABLE: 0 = NO 1 = YES
9:0	0x3ff	OVERFLOW_TH

25.9.3 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 bringup; a feature that SW shouldn't worry about.

LFSR_COUNT: LFSR_COUNT is the rate of filling LFSR data into the FIFO. The rate needs to be close to actual spdif 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 = 277Mhz:

- 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 FIFO when LFSR_DATA_EN is set.

VALIDITY: This is the validity bit that is going into FIFO when LFSR_DATA_EN is set.

LFSR_DATA_EN: When this bit is set, FIFO will be filled with LFSR (Linear Feedback Shift Register) data and ignore incoming spdif input stream. LFSR will generate pseudorandom data in the following pattern:

F->E->C->8->1->2->4->9->3->6->D->A->5->B->7->F->E.....

Offset: 081h l Read/Write: R/W l Reset: 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

25.9.4 HDMI_NV_PDISP_AUDIO_FS1_0

When a B preamble (A B preamble is an indicator of the start of a frame in SPDIF) 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->7, one entry per audio frequency)

For one 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 frequncy 'x' in the AUDIO_CNTRL0_SAMPLING_FREQ register field.

- FS1 is for fs = 32.0KHz.
- FS2 is for fs = 44.1KHz.
- FS3 is for fs = 48.0KHz.
- FS4 is for fs = 88.2KHz.
- FS5 is for fs = 96.0KHz.
- FS6 is for fs = 176.4KHz.
- FS7 is for fs = 192.0KHz.

The formula to compute 8 HALFs is following:

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 32kHz and dispclk_audio_fs is 216 Mhz, then the 8 HALFs (or 4 bits) preamble length is:

8 HALFs = 8 * 216.0Mhz / (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 32kHz of audio sampling frequency is detected.

The init values of the registers below are based on a hdmi_audio_clk of 216 MHz, and SW should reprogram these every time that changes (which should not happen.)

AUDIO_FS1

LOW: Low threshold (in dispclk periods) for 32kHz audio detection.

HIGH: High threshold (in dispclk periods) for 32kHz audio detection.

Offset: 082h l Read/Write: R/W l Reset: 0b000110101110xxxx000110011100

Bit	Reset	Description
27:16	0x1ae	HIGH
11:0	0x19c	LOW

25.9.5 HDMI_NV_PDISP_AUDIO_FS2_0

AUDIO_FS2

LOW: Low threshold (in dispclk periods) for 44.1kHz audio detection.

HIGH: High threshold (in dispclk periods) for 44.1kHz audio detection.

Offset: 083h l Read/Write: R/W l Reset: 0b000100111011xxxx000100101001

Bit	Reset	Description
27:16	0x13b	HIGH
11:0	0x129	LOW

25.9.6 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: 084h l Read/Write: R/W l Reset: 0b000100100010xxxx000100010000

Bit	Reset	Description
27:16	0x122	HIGH
11:0	0x110	LOW

25.9.7 HDMI_NV_PDISP_AUDIO_FS4_0

AUDIO_FS4

LOW: Low threshold (in dispclk periods) for 88.2kHz audio detection.

HIGH: High threshold (in dispclk periods) for 88.2kHz audio detection.

Offset: 085h l Read/Write: R/W l Reset: 0b000010011111xxxx000010010011

Bit	Reset	Description
27:16	0x9f	HIGH
11:0	0x93	LOW

25.9.8 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: 086h l Read/Write: R/W l Reset: 0b000010010010xxxx000010000110

Bit	Reset	Description
27:16	0x92	HIGH
11:0	0x86	LOW

25.9.9 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: 087h l Read/Write: R/W l Reset: 0b000001001110xxxx000001001010

Bit	Reset	Description
27:16	0x4e	HIGH
11:0	0x4a	LOW

25.9.10 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: 088h l Read/Write: R/W l Reset: 0b000001001000xxxx000001000100

Bit	Reset	Description
27:16	0x48	HIGH
11:0	0x44	LOW

25.9.11 HDMI_NV_PDISP_AUDIO_PULSE_WIDTH_0

AUDIO_PULSE_WIDTH

This is a status only register used for debugging.

HALF: Because 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.

e.g. 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 spdif input stream. THREE_HALF should be approximately 3 times of HALF, this only happen in 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: 089h l Read/Write: R/W l Reset: 0b0xx01xxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
28	RW	0x0	LOCK: 0 = NO 1 = YES
25:24	RW	0x1	COUNT2LOCK
21:12	RO	X	THREE_HALF
9:0	RO	X	HALF

25.9.12 HDMI_NV_PDISP_AUDIO_THRESHOLD_0

AUDIO_THRESHOLD

This is a status only register which is useful for debugging. It provides information about some characteristics of the SPDIF stream.

THRESHOLD_LOW: THRESHOLD_LOW is 1.5 times of HALF. If a spdif 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 spdif pulse is less than or equal to THRESHOLD_HIGH and bigger than THRESHOLD_LOW, then a bit '0' is detected.

Offset: 08ah l Read/Write: RO l Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
21:12	X	HIGH
9:0	X	LOW

25.9.13 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 spdif 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 = 416Mhz. It is anticipated that SW doesn't 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 HW 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 reports the incoming spdif audio stream sampling frequency HDMI spec only support 7 audio sample frequency (fs), anything other than those 7 fs, it will be reported as UNKNOWN.

SOURCE_SELECT: Determines whether to use the SPDIF or Azalia (HDAL) audio input. This field has no effect on chips without the HDAL input. The default is AUTO: selects SPDIF 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 spec each block has 192 (0xC0) frames.

Offset: 08bh l Read/Write: R/W l Reset: 0b11000000xxxxxxxxxx0xxx000000110

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 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

25.9.14 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 spec.

N_RESET: N_RESET is reset for N counter, If software is controlling the value of N, every time the audio stream changes sampling freq (fs), software (RM) need to reset the N counter by writing _ASSERT to N_RESET and follow 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 spec Chapter 7. This bit is strictly for debugging purpose only.

_NORMAL: This method increments the CTS counter a variable number of times based on the length of the SPDIF 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: 08ch l Read/Write: R/W l Reset: 0b0xxx1xxx00000000000000000000000

Bit	Reset	Description
28	0x0	LOOKUP: 1 = ENABLE 0 = DISABLE
24	0x1	GENERATE: 0 = NORMAL 1 = ALTERNATE
20	0x0	RESETF: 1 = ASSERT 0 = DEASSERT
19:0	0x0	VALUE

25.9.15 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 32kHz audio at the current pixel clock frequency should be written here.

Offset: 08dh l Read/Write: R/W l Reset: 0b00000001000000000000

Bit	Reset	Description
19:0	0x1000	VAL_0320_VALUE

25.9.16 HDMI_NV_PDISP_AUDIO_NVAL_0441_0

VALUE: The correct N value for 44.1kHz audio at the current pixel clock frequency should be written here.

Offset: 08eh l Read/Write: R/W l Reset: 0b00000001100010000000

Bit	Reset	Description
19:0	0x1880	VAL_0441_VALUE

25.9.17 HDMI_NV_PDISP_AUDIO_NVAL_0882_0

VALUE: The correct N value for 88.2kHz audio at the current pixel clock frequency should be written here.

Offset: 08fh l Read/Write: R/W l Reset: 0b00000011000100000000

Bit	Reset	Description
19:0	0x3100	VAL_0882_VALUE

25.9.18 HDMI_NV_PDISP_AUDIO_NVAL_1764_0

VALUE: The correct N value for 176.4kHz audio at the current pixel clock frequency should be written here.

Offset: 090h l Read/Write: R/W l Reset: 0b00000110001000000000

Bit	Reset	Description
19:0	0x6200	VAL_1764_VALUE

25.9.19 HDMI_NV_PDISP_AUDIO_NVAL_0480_0

VALUE: The correct N value for 48kHz audio at the current pixel clock frequency should be written here.

Offset: 091h l Read/Write: R/W l Reset: 0b00000001100000000000

Bit	Reset	Description
19:0	0x1800	VAL_0480_VALUE

25.9.20 HDMI_NV_PDISP_AUDIO_NVAL_0960_0

VALUE: The correct N value for 96kHz audio at the current pixel clock frequency should be written here.

Offset: 092h l Read/Write: R/W l Reset: 0b00000011000000000000

Bit	Reset	Description
19:0	0x3000	VAL_0960_VALUE

25.9.21 HDMI_NV_PDISP_AUDIO_NVAL_1920_0

VALUE: The correct N value for 192kHz audio at the current pixel clock frequency should be written here.

Offset: 093h l Read/Write: R/W l Reset: 0b00000110000000000000

Bit	Reset	Description
19:0	0x6000	VAL_1920_VALUE

25.9.22 HDMI_NV_PDISP_HDCPRIF_ROM_TIMING_0

This register controls the timing of the cryptorom interface. Any time the hdmi_clk period changes, the values here must be changed too.

The cryptorom bit period must more than 1 uS. 500 KHz is a safe target. The hdmi_clk frequency is reduced by the scale factor defined by the PRESCALE divider; the period is equal to (PRESCALE+1) hdmi_clk ticks.

BIT_PERIOD defines the timing of the SCL (serial clock) output. The SCL pin will be driven high for counts 0 .. (BIT_PERIOD >> 1), and will be driven low for counts (BIT_PERIOD >> 1)+1 .. BIT_PERIOD.

DATA_DLY defines how many scaled ticks pass before SDA (the data bit) makes a transition after the falling edge of SCL (for data writes and for ACK'ing after data reads).

START_DLY defines how many scaled ticks pass before SDA (the data bit) makes a transition after the rising edge of SCL (for START and STOP).

The defaults make sure even with an HDMI clock of 150 MHz, the bit clock will not be faster than 500 KHz.

In general, programming should be something like this. Given the hdmi_clk frequency and the target I2C bit rate, figure out the necessary prescale. The prescale must be large enough such that the BIT_PERIOD divisor fits into 8 bits, but it should otherwise be as small as possible in order to minimize the time quantization.

(PRESCALE+1)*(BIT_PERIOD+1)*(hdmi_clk period) >= (I2C bit period)

BIT_PERIOD+1 can be as large as 256. Rearranging:

PRESCALE = ceil[(I2C bit period)/(hdmi_clk period) / 256] - 1

= ceil[(hdmi_clk frequency)/(I2C frequency) / 256] - 1

This determines the time quantum of the other divisors:

- prescale_period = (hdmi_clk period) / (PRESCALE + 1)

Next, determine the bit period:

- BIT_PERIOD = ceil[(hdmi_clk frequency) / (PRESCALE+1) / (I2C frequency)] - 1

SDA (data) normally changes only while SCL is low. The Atmel datasheet says that SDA must hold for at least 10ns after SCL falls and must be set up at least 100 ns before SCL rises. Thus, DATA_DLY must be:

- DATA_DLY > floor(BIT_PERIOD/2) + 10 ns
- DATA_DLY < BIT_PERIOD - 100 ns

In most circumstances setting DATA_DLY to arrive 75% of the way through the cycle is OK.

- DATA_DLY = floor((BIT_PERIOD+1) * 3 / 4) - 1

The start and stop conditions extend the SCL high time before the next falling SCL transition. One Atmel datasheet says that SDA (data) must not change within 200 ns of the rising or falling edge of SCL. A safe calculation is to assume that we wait a tick less than half a clock period:

- START_DLY = floor(BIT_PERIOD / 2) - 2

Offset: 094h l Read/Write: R/W l Reset: 0b11110000100000001110000010011

Bit	Reset	Description
27:24	0xf	PRESCALE
23:16	0x8	START_DLY: used for start and stop timing
15:8	0xe	DATA_DLY

Bit	Reset	Description
7:0	0x13	BIT_PERIOD

25.9.23 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 us time reference, otherwise the time intervals requested by NV_PDISP_SOR_SEQ_INSTn won't 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 us" 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 don't need to be exact. If the integer part is written as 0, it will be interpreted the same as "1".

Offset: 095h l Read/Write: R/W l Reset: 0b0001100100

Bit	Reset	Description
15:8	0x19	DIV_INT: default: 27 MHz
7:6	0x0	DIV_FRAC

25.9.24 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: 096h l Read/Write: R/W l Reset: 0b0

Bit	Reset	Description
0	NO	ARM_CRC_ENABLE: 0 = NO 1 = YES 0 = DIS 1 = EN

25.9.25 HDMI_NV_PDISP_INPUT_CONTROL_0

INPUT_CONTROL

HDMI_SRC_SELECT selects which of the two display units to take input from. 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 EIA/CEA-861-B and HDMI specs, 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 HDMI 1.2a spec section 6.6.

This register is double-buffered, and will take effect on next frame boundary.

Offset: 097h l Read/Write: R/W l Reset: 0b00

Bit	Reset	Description
1	FULL	ARM_VIDEO_RANGE: 0 = FULL 1 = LIMITED
0	DISPLAY	HDMI_SRC_SELECT: 0 = DISPLAY 1 = DISPLAYB

25.9.26 HDMI_NV_PDISP_SCRATCH_0

SCRATCH

This register is not used by hardware. It is available for software to store state, or for testing purposes.

Offset: 098h l Read/Write: R/W l Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	DATA

25.9.27 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)

- 0000 -> 0.0mA
- 0001 -> 0.5mA
- 0010 -> 1.0mA
- 0011 -> 1.5mA
- 0100 -> 2.0mA
- 0101 -> 2.5mA
- 0110 -> 3.0mA
- 0111 -> 3.5mA
- 1000 -> 4.0mA
- 1001 -> 4.5mA
- 1010 -> 5.0mA
- 1011 -> 5.5mA
- 1100 -> 6.0mA

- 1101 -> 6.5mA
- 1110 -> 7.0mA
- 1111 -> 7.5mA

see related pre-emphasis control registers above in NV_PDISP_SOR_PLL0 register

Offset: 099h l Read/Write: R/W l Reset: 0b1010xxxx1010xxxx1010xxxx1010

Bit	Reset	Description
27:24	0xa	PE_CURRENT3
19:16	0xa	PE_CURRENT2
11:8	0xa	PE_CURRENT1
3:0	0xa	PE_CURRENT0

25.9.28 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 autoincrement 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 the 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 autoincrement 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, but note that only PKEY_LOADED will indicated 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: 1'st downstream key
- Byte 12 to Byte 18: 2'nd downstream key

  ...
- Byte 278 to Byte 284: 40'th downstream key
- Byte 285 to Byte 289: upstream KSV
- Byte 290 to Byte 296: 1'st upstream key

  ...
- Byte 563 to Byte 569: 40'th upstream key

Offset: 09ah l Read/Write: R/W l Reset: 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

25.9.29 HDMI_NV_PDISP_KEY_DEBUG0_0

Offset: 09bh l Read/Write: R/W l Reset: 0bxx0xxx0

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

25.9.30 HDMI_NV_PDISP_KEY_DEBUG1_0

Offset: 09ch l Read/Write: R/W l Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	CHECKSUMVAL_HIGH
15:0	0x0	CHECKSUMVAL_LOW

25.9.31 HDMI_NV_PDISP_KEY_DEBUG2_0

Offset: 09dh l Read/Write: R/W l Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxx0xx0

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

25.9.32 HDMI_NV_PDISP_KEY_HDCP_KEY_0_0

Offset: 09eh l Read/Write: WO l Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	KEY_REG

25.9.33 HDMI_NV_PDISP_KEY_HDCP_KEY_1_0

Offset: 09fh l Read/Write: WO l Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	KEY_REG

25.9.34 HDMI_NV_PDISP_KEY_HDCP_KEY_2_0

Offset: 0a0h l Read/Write: WO l Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	KEY_REG

25.9.35 HDMI_NV_PDISP_KEY_HDCP_KEY_3_0

Offset: 0a1h l Read/Write: WO l Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	KEY_REG

25.9.36 HDMI_NV_PDISP_KEY_HDCP_KEY_TRIG_0

Offset: 0a2h l Read/Write: R/W l Reset: 0b0

Bit	Reset	Description
8	IDLE	LOAD_HDCP_KEY: 0 = IDLE 1 = TRIGGER

25.9.37 HDMI_NV_PDISP_KEY_SKEY_INDEX_0

16 sets of AES keys are available. Convention is:

15 : test/debug

0-14: production

Offset: 0a3h l Read/Write: WO l Reset: 0b0000

Bit	Reset	Description
3:0	0x0	IDX_VALUE: 15 = TEST

25.9.1 HDMI_NV_PDISP_SOR_AUDIO_CNTRL0_0

HD Audio (aka Azalia) audio

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 reports the incoming audio stream sampling frequency in the Azalia codec. The HDMI spec only supports 7 audio sample frequency (fs). Anything other than those 7 fs will be reported as UNKNOWN. See HDMI 1.1 spec Table 7-4 (page 77)

SOURCE_SELECT: Determines whether to use the SPDIF or Azalia (HDAL) audio input. This field has no effect on chips without the HDAL input. The default is AUTO: selects SPDIF 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 Azaalia data, SPDIF is SPDIF data.

Offset: 0ach | Read/Write: R/W | Reset: 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

25.9.2 HDMI_NV_PDISP_SOR_AUDIO_DEBUG_0

This register is used for debug purposes only.

FIFO_ERROR: The data is passed from the audio (azalia or spdif) 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 of sor units does not drain away the audio sample data fast enough.

Offset: 0adh | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	NO	FIFO_ERROR: 0 = NO 1 = YES

25.9.3 HDMI_NV_PDISP_SOR_AUDIO_SPARE0_0

This register is a backup register.

Offset: 0aeh | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	Reserved

25.9.4 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0320_0

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) will always be used.

VALUE: The correct N value for 32kHz audio at the current pixel clock frequency should be written here.

Offset: 0afh | Read/Write: R/W | Reset: 0b00000001000000000000

Bit	Reset	Description
19:0	0x1000	VALUE

25.9.5 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0441_0

VALUE: The correct N value for 44.1kHz audio at the current pixel clock frequency should be written here.

Offset: 0b0h | Read/Write: R/W | Reset: 0b00000001100010000000

Bit	Reset	Description
19:0	0x1880	VALUE

25.9.6 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0882_0

VALUE: The correct N value for 88.2kHz audio at the current pixel clock frequency should be written here.

Offset: 0b1h | Read/Write: R/W | Reset: 0b00000011000100000000

Bit	Reset	Description
19:0	0x3100	VALUE

25.9.7 HDMI_NV_PDISP_SOR_AUDIO_NVAL_1764_0

VALUE: The correct N value for 176.4kHz audio at the current pixel clock frequency should be written here.

Offset: 0b2h | Read/Write: R/W | Reset: 0b00000110001000000000

Bit	Reset	Description
19:0	0x6200	VALUE

25.9.8 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0480_0

VALUE: The correct N value for 48kHz audio at the current pixel clock frequency should be written here.

Offset: 0b3h | Read/Write: R/W | Reset: 0b00000001100000000000

Bit	Reset	Description
19:0	0x1800	VALUE

25.9.9 HDMI_NV_PDISP_SOR_AUDIO_NVAL_0960_0

VALUE: The correct N value for 96kHz audio at the current pixel clock frequency should be written here.

Offset: 0b4h | Read/Write: R/W | Reset: 0b00000011000000000000

Bit	Reset	Description
19:0	0x3000	VALUE

25.9.10 HDMI_NV_PDISP_SOR_AUDIO_NVAL_1920_0

VALUE: The correct N value for 192kHz audio at the current pixel clock frequency should be written here.

Offset: 0b5h | Read/Write: R/W | Reset: 0b00000110000000000000

Bit	Reset	Description
19:0	0x6000	VALUE

25.9.11 HDMI_NV_PDISP_SOR_AUDIO_HDA_SCRATCH0_0

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: 0b6h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	RESERVED

25.9.12 HDMI_NV_PDISP_SOR_AUDIO_HDA_SCRATCH1_0

Offset: 0b7h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	RESERVED

25.9.13 HDMI_NV_PDISP_SOR_AUDIO_HDA_SCRATCH2_0

Offset: 0b8h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	RESERVED

25.9.14 HDMI_NV_PDISP_SOR_AUDIO_HDA_SCRATCH3_0

Offset: 0b9h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description

Bit	Reset	Description
31:0	0x0	RESERVED

25.9.15 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 RM if that functionality is ever needed.

When bit 31 changes, an interrupt can be generated (see INT_STATUS register)

Offset: 0bah | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DATA

25.9.16 HDMI_NV_PDISP_SOR_AUDIO_HDA_CODEC_SCRATCH1_0

Offset: 0bbh | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	DATA

25.9.17 HDMI_NV_PDISP_SOR_AUDIO_HDA_ELD_BUFWR_0

SW 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 (Modified from Intel DCN HDA-034-A)

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 HW 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 RM and read by the audio driver. Once the ELD buffer is written by RM, the valid bit, NV_PDISP_SOR_AUDIO_HDA_PRESENSE_ELDV is set to _VALID to indicate that ELD contents have been initialized by RM. On hot unplug events, NV_PDISP_SOR_AUDIO_HDA_PRESENSE_ELDV is set to _INVALID to erase ELD programming in the audio driver.

Offset: 0bch | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:8	0x0	INDEX
7:0	0x0	DATABYTE

25.9.18 HDMI_NV_PDISP_SOR_AUDIO_HDA_PRESENSE_0

Reports the HotPlug 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: Presense Detect. This should be set to _PRESENT by SW upon hotplug and _NOT_PRESENT on unplug.

Offset: 0bdh | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
1	INVALID	ELDV: 0 = INVALID 1 = VALID
0	NOT_PRESENT	PD: 0 = NOT_PRESENT 1 = PRESENT

25.9.19 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: 0beh | Read/Write: RO | Reset: 0bxxx

Bit	Reset	Description
2	X	REQUEST_STATE_VALID
1:0	X	REQUEST_STATE: 0 = DONT_CARE 2 = PROTECTION_OFF 3 = PTOTECTION_ON

25.9.20 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 SPDIF, the N counter will run at 128*fs (audio sampling frequency) and count to a value determined by the HDMI spec. The Azalia counter will run at 24MHz at all times, and it needs to count for the same period of time as the N counter would have.

Refer section 7.2 of HDMI spec 1.2

AVAL values do not need to be programmed by SW. 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, clock is 24 MHz, not 54 MHz. For 44.1, default N of 6272 gives non-integer AVAL. Choosing N of 4704 instead (nominal CTS 61875) gives 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 we hard-code CTS/N 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 32kHz audio at the current pixel clock frequency should be written here.

Offset: 0bfh | Read/Write: R/W | Reset: 0b00000101110110111000000

Bit	Reset	Description
19:0	0x5dc0	VALUE

25.9.21 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0441_0

VALUE: The correct A value for 44.1kHz audio at the current pixel clock frequency should be written here.

Offset: 0c0h | Read/Write: R/W | Reset: 0b00000100111000100000

Bit	Reset	Description
19:0	0x4e20	VALUE

25.9.22 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0882_0

VALUE: The correct A value for 88.2kHz audio at the current pixel clock frequency should be written here.

Offset: 0c1h | Read/Write: R/W | Reset: 0b00000100111000100000

Bit	Reset	Description
19:0	0x4e20	VALUE

25.9.23 HDMI_NV_PDISP_SOR_AUDIO_AVAL_1764_0

VALUE: The correct A value for 176.4kHz audio at the current pixel clock frequency should be written here.

Offset: 0c2h | Read/Write: R/W | Reset: 0b00000100111000100000

Bit	Reset	Description
19:0	0x4e20	VALUE

25.9.24 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0480_0

VALUE: The correct A value for 48kHz audio at the current pixel clock frequency should be written here.

Offset: 0c3h | Read/Write: R/W | Reset: 0b00000101110110111000000

Bit	Reset	Description
19:0	0x5dc0	VALUE

25.9.25 HDMI_NV_PDISP_SOR_AUDIO_AVAL_0960_0

VALUE: The correct A value for 96kHz audio at the current pixel clock frequency should be written here.

Offset: 0c4h | Read/Write: R/W | Reset: 0b00000101110110111000000

Bit	Reset	Description
19:0	0x5dc0	VALUE

25.9.26 HDMI_NV_PDISP_SOR_AUDIO_AVAL_1920_0

VALUE: The correct A value for 192kHz audio at the current pixel clock frequency should be written here.

Offset: 0c5h | Read/Write: R/W | Reset: 0b00000101110110111000000

Bit	Reset	Description
19:0	0x5dc0	VALUE

25.9.27 HDMI_NV_PDISP_SOR_AUDIO_AVAL_DEFAULT_0

VALUE: Default A value if aza codec sampling frequency doesn't match any of the above

Offset: 0c6h | Read/Write: R/W | Reset: 0b00000101110110111000000

Bit	Reset	Description
19:0	0x5dc0	VALUE

25.9.28 HDMI_NV_PDISP_SOR_AUDIO_GEN_CTRL_0

This register only takes effect when it is written by software. The init 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: 0c7h | Read/Write: R/W | Reset: 0b00000000001000000xxxxxxxx00000001

Bit	Reset	Description
31:16	0x20	DEV_ID
7:0	0x1	REV_ID

25.9.29 HDMI_NV_PDISP_INT_STATUS_0

Sticky interrupt status, write 1 to clear

Interrupt support: Each interrupt event has three configuration states:

MASK	ENABLE	result
x	0	nothing
0	1	INT_STATUS asserted
1	1	INT_STATUS asserted, and interrupt pin asserted

Offset: 0cch | Read/Write: R/W | Reset: 0bxxxx

Bit	Reset	Description
3	X	SCRATCH: SW has written NV_PDISP_SCRATCH register (bit 31 change)
2	X	CP_REQUEST: HDA Codec has written CP_REQUEST register
1	X	CODEC_SCRATCH1: HDA Codec has written CODEC_SCRATCH1 register (bit 31 change)
0	X	CODEC_SCRATCH0: HDA Codec has written CODEC_SCRATCH0 register (bit 31 change)

25.9.30 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: 0cdh | Read/Write: R/W | Reset: 0b0000

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

25.9.31 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: 0ceh | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3	DISABLE	SCRATCH_ENABLE: 0 = DISABLE 1 = ENABLE
2	DISABLE	CP_REQUEST_ENABLE: 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
1	DISABLE	CODEC_SCRATCH1_ENABLE: 0 = DISABLE 1 = ENABLE
0	DISABLE	CODEC_SCRATCH0_ENABLE: 0 = DISABLE 1 = ENABLE

[THIS PAGE INTENTIONALLY LEFT BLANK]

26.0 MIPI-CSI (CAMERA SERIAL INTERFACE)

The Camera Serial Interface (CSI) is based on MIPI CSI 2.0 standard specification and implements the CSI receiver which receives data from an external camera module with a CSI transmitter. It consists of two CSI receiver interfaces so it can receive serial transmission from two cameras:

- CSI-A interface supports 1-clock lane and up to 4 data lanes. In 3 and 4-data lane configurations, CSI-A uses CSI-B's data lanes.
- CSI-B interface supports 1 clock lane and 2-data lanes.

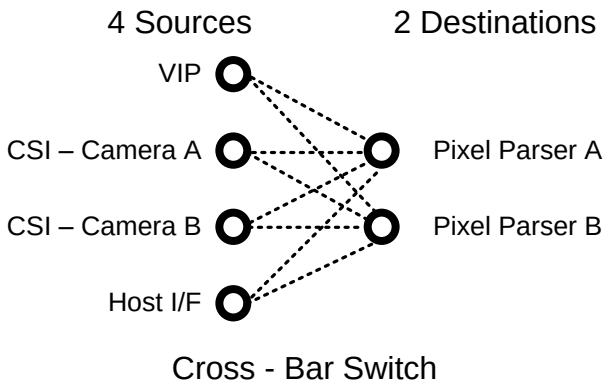
The CSI interface can also receive a digital CSI compatible byte stream from either the 8-bit Video Input (VI) port or from the host interface. The CSI stream coming from VI port is provided to accommodate the possibility of using an external CSI receiver or to take advantage of the rich CSI data formats transmission using the traditional 8-bit digital CMOS video interface. Similarly, CSI stream coming from host is provided to take advantage of the rich CSI data formats.

Only 1 data/pixel stream can be processed at any given time. The two streams can come from any one or two of the four possible sources, as shown in Figure 66. 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. Since there are only two pixel parsers, virtual channel and embedded data capability cannot be used at the same time.

CSI supports both single-shot mode and continuous mode.

CSI is designed with error resilience.

Figure 66: Data Source/Destination Options



The MIPI CSI 2.0 standard is available from MIPI to its members at http://www.mipi.org/

26.1 Use Cases

The use cases for CSI fall into the following categories listed in Table 71:

Table 71: CSI Use Cases

Case	Description	Sources	Destinations
Normal	Standard CSI	Camera A or B	PPA or PPB
Virtual Channel	One stream goes to 2 parsers. Packet filtering based on VC	Camera A,B	PPA and PPB

Case	Description	Sources	Destinations
	number in header	VIP, or Host	
Embedded Data	One stream goes to 2 parsers. Packet filtering based on data type in header	Camera A,B VIP, or Host	PPA and PPB
Embedded Data (without filtering)	One stream goes to 1 parser. Has normal and embedded data	Camera A,B VIP, or Host	PPA or PPB
VIP	CSI payload-only or packet sources from VIP (Diagnostic Workaround mode) *	VIP	PPA or PPB
Host	Payload-only or Packet sources from Host I/F	Host I/F	PPA or PPB

* In the Tegra® 3 devices, packet mode in the CSI VIP path is not supported.

26.2 Input Data Format

The supported data formats are summarized in Table 72.

Table 72: Supported CSI Formats

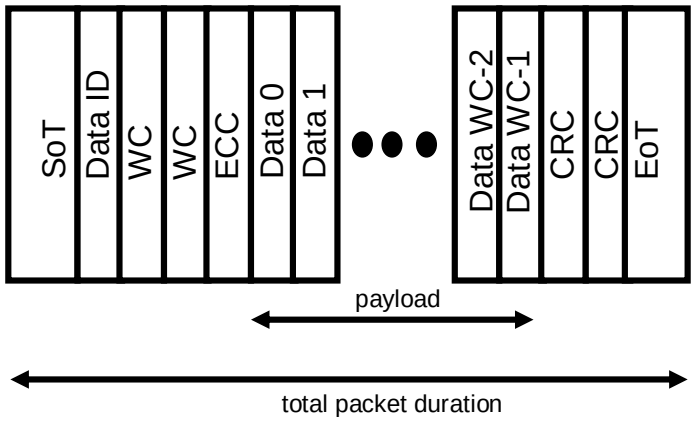
Data Format	
RGB	RGB888, RGB666, RGB565, RGB555, RGB444
YUV	YUV422-8b, YUV422-10b, YUV420-8b (legacy), YUV420-8b, YUV420-10b, YUV444-8b
RAW	RAW6, RAW7, RAW8, RAW10, RAW12, RAW14
User defined	JPEG8
Embedded	Embedded control information

The formats YUV422/420-10b are converted to YUV422/420-8b by taking the most significant bits.

26.3 CSI Packet Structure

The CSI packet structure is illustrated in Figure 67. 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 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 LP11 state.

Figure 67: CSI Packet Structure



26.4 CSI Implementation

The CSI interface essentially consists of:

- MIPI D-Phy block
- Control Interface Logic (CIL) for Serial Clock Receiver
- CSI data path module which is implemented as a sub-module of VI module

In Tegra 3 implementations, 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 8b 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 8b packet aligned data from CSICIL, performs parity checking on the header, header decoding, CRC check, and pixel parsing. The output is sent out to VI through the 28b bus which outputs 1 pixel per VI clock.

26.5 Performance Limitations

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 data paths. This CSI byte clock is derived from the received CSI serial clock by dividing the CSI serial clock by 4. So for 1Gbps 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). The maximum output pixel clock rate of CSI module is 300 MHz.

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 ISP, VI and EPP specification.

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 73: Packet Efficiency

	Best* WC=65536	Typical* WC=1024
Payload	524us	8192ns
Overhead	356ns	356ns
Efficiency	99.9%	96%

26.6 Error Resilience

CSI is required to be designed to tolerate and/or recover stream errors at various levels.

According to the CSI 2.0 specification, 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, CSI has to always send “valid” stream data to VI, even when the incoming stream contains errors and causes 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.

26.7 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.
- CSI stream from VI port is limited to VI port input clock frequency (96 MHz) at 8-bit/clock. This stream may have fully compatible CSI packets format or it may carry only CSI compatible data format without packet header/footer and short packets. If it carries only CSI compatible data format, it should be transmitted using VI video input stream format with external H/V syncs.
- CSI stream from host interface should be sent in 32-bit data through the Y-FIFO with frame/line definition compatible only. So, CSI stream coming from host should only contain CSI compatible data formats but without the CSI packet header/footer or short packets. The header information should be set using registers.
- Same stream going to different pixel parsers. This case will work as long as no stall comes from VI side. If stall comes then the current frame will be invalid. This is an exception case which will never normally occur. STM_ERR will be generated and S/W needs to take appropriate action.

26.8 CSI Datapath Module

The CSI datapath consists of four input ports and two output ports. There are two datapaths for pixel stream processing; therefore, at any time, a maximum of two input ports maybe active simultaneously.

The components of CSI datapath include:

- Four input ports from: CSI A interface, CSI B interface, 8-bit VI port, and 32-bit host interface. Each port is capable of carrying a stream with CSI compatible data format.
- Three asynchronous input buffers (FIFOs) to receive streams from VI port, CSI A interface, and CSI B interface.
- Four 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 CSI packet data stream to an output a pixel stream. Each output port consists of maximum 24-bit of data per clock. Depending on the output data format options, one or two pixels per clock may be sent.

26.8.1 Header Parser

CSI pixel stream processing mainly consists of header parser and pixel parser. There are 4 header parsers: header parser A, header parser B, header parser V, 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 VI or host, there is an option to indicate Frame Start using the incoming vertical sync signal. If vertical sync signal is used to indicate frame start then if 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 make preparation to process a new frame and output frame start code.

26.8.1.1 Data Type

There are 64 possible types based on 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

26.8.1.2 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

26.8.1.3 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 Section 26.2 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 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 spec, 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 CSI 2.0 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 as 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.

26.8.1.4 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.

26.9 Software Requirements

26.9.1 Error Counters

To help debug, three 32-bit error counters have been implemented. SW 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

26.9.2 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
2. Initialize and power up CSI interface
3. Wait for initialization time or done signal from calibration logic
4. Power up camera thru I2C interface
5. All CSI data and clock lanes are in stop state, LP11
6. Initiate frame capture thru I2C
7. Frame done, CSI goes back to stop state, LP11

26.9.3 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 put camera in sleep mode thru I2C
2. Camera sends ULPM command to CSI
3. CSI receives ULPM and raise interrupt
4. Software powers down CSI which goes to sleep in sync with camera’s CSI transmitter

26.10 DPHY Modes of Operation

The MIPI DPHY has 3 basic modes of operation. High speed mode employs differential signaling and achieves data rate of 1Gbps. Both the driver and receiver are matched to 100 ohm 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. In the NV implementation, 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 20Mbps for a 50ns 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 20MHz.

Table 74: MIPI DHY Mode of Operations

Modes	Description	Clock
High speed (HS)	High speed differential signaling. Upto 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

26.11 MIPI-CSI Registers

26.11.1 CSI_VI_INPUT_STREAM_CONTROL_0

VI Input Stream Control

Offset: 200h | Read/Write: R/W | Reset: 0bx

Bit	Reset	Description
7	X	VIP_SF_GEN: VIP Start Frame Generation Don't 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 3 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

26.11.2 CSI_HOST_INPUT_STREAM_CONTROL_0

Host Input Stream Control

Offset: 202h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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 Don't 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 a SF control,

Bit	R/W	Reset	Description
			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

26.11.3 CSI_INPUT_STREAM_A_CONTROL_0

CSI Input Stream A Control

Offset: 204h | Read/Write: R/W | Reset: 0b011111111xxxxxxxxxxx0xx01

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
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

26.11.4 CSI_PIXEL_STREAM_A_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 206h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000 |

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 fields 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

Bit	Reset	Description
		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
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) 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 maybe 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 maybe 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 MSB of the CSI Data Identifier (DI) byte. This value will be ignored if the source stream doesn't 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 LSB of the CSI Data Identifier (DI) byte. If the source stream doesn't 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

Bit	Reset	Description
		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_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

26.11.5 CSI_PIXEL_STREAM_A_CONTROL1_0

CSI Pixel Stream A Control 1

Offset: 207h | Read/Write: R/W | Reset: 0b00000000 |

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 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.

26.11.6 CSI_PIXEL_STREAM_A_WORD_COUNT_0

CSI Pixel Stream A Word Count

Offset: 208h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx |

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

26.11.7 CSI_PIXEL_STREAM_A_GAP_0

CSI Pixel Stream A Gap

Offset: 209h | 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 is to ensure that 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 is to ensure that 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.

26.11.8 CSI_PIXEL_STREAM_PPA_COMMAND_0

CSI Pixel Parser A Command

Offset: 20ah | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx00

Bit	Reset	Description
15:12	X	CSI_PPA_START_MARKER_FRAME_MAX: CSI Pixel Parser A Start Marker Maximum Start Frame is indicated when Min 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 Max 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 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_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 SW 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 Pixel parser is enabled later than the corresponding input source, CSI will keep on rejecting incoming stream, till 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

26.11.9 CSI_PIXEL_STREAM_A_EXPECTED_FRAME_0

CSI Pixel Stream A Expected Frame

Offset: 232h | 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 outputted 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 outputted, when the fake EF is generated is less than the expected frame height. The type of padding is specified using CSI_PPA_PAD_FRAME.

26.11.10 CSI_INPUT_STREAM_B_CONTROL_0

CSI Input Stream B Control

Offset: 20fh | Read/Write: R/W | Reset: 0b01111111xxxxxxxxxxx0xx01

Bit	Reset	Description
23:16	0x7f	CSI_B_SKIP_PACKET_THRESHOLD: CSI-B 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_B_SKIP_PACKET_THRESHOLD_ENABLE: Enables skip packet threshold feature. Skip packet feature is enabled. 0 = DISABLE : Skip packet feature is disabled. 1 = ENABLE
1:0	0x1	CSI_B_DATA_LANE: CSI-B Data Lane 0= 1 data lane 1= 2 data lanes 2= 3 data lanes (not supported on SC17 & SC25) 3= 4 data lanes (not supported on SC17 & SC25)

26.11.11 CSI_PIXEL_STREAM_B_CONTROL0_0

CSI Pixel Stream A Control 0

Offset: 211h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxx000

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 fields 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

Bit	Reset	Description
		value if this fields 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
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 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 maybe 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 maybe 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 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_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 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 MSB of the CSI Data Identifier (DI) byte. This value will be ignored if the source stream doesn't contain packet headers, or CSI_PPB_DATA_IDENTIFIER is DISABLED, then this value will be ignored. 0 = ONE 1 = TWO 2 = THREE 3 = FOUR
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 LSB of the CSI Data Identifier (DI) byte. If the source stream doesn't 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

Bit	Reset	Description
		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
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

26.11.12 CSI_PIXEL_STREAM_B_CONTROL1_0

CSI Pixel Stream B Control 1

Offset: 212h | Read/Write: R/W | Reset: 0b00000000

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.

26.11.13 CSI_PIXEL_STREAM_B_WORD_COUNT_0

CSI Pixel Stream A Word Count

Offset: 213h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx

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 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

26.11.14 CSI_PIXEL_STREAM_B_GAP_0

CSI Pixel Stream B Gap

Offset: 214h | 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 is to ensure that 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 is to ensure that 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.

26.11.15 CSI_PIXEL_STREAM_PPB_COMMAND_0

CSI Pixel Parser B Command

Offset: 215h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxx00

Bit	Reset	Description
15:12	X	CSI_PPB_START_MARKER_FRAME_MAX: CSI Pixel Parser B Start Marker Maximum Start Frame is indicated when Min 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 Max 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 who's 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 SW should Clear it along with disabling the CSI_PPB_ENABLE, once a frame is captured 0 = DISABLE 1 = ENABLE
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 Pixel parser is enabled later than the corresponding input source, csi will keep on rejecting incoming stream, till 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

26.11.16 CSI_PIXEL_STREAM_B_EXPECTED_FRAME_0

CSI Pixel Stream B Expected Frame

Offset: 233h | 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 outputted 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 outputted, when the fake EF is generated is less than the expected frame height. The type of padding is specified using CSI_PPB_PAD_FRAME.

26.11.17 CSI_PHY_CIL_COMMAND_0

CSI Phy and CIL Command

Offset: 21ah | Read/Write: R/W | Reset: 0b00xxxxxxxxxxxxx00

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. disable (reset) 0 = NOP : no operation 1 = ENABLE : enable 2 = DISABLE
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. disable (reset) 0 = NOP : no operation 1 = ENABLE : enable 2 = DISABLE

26.11.18 CSI_PHY_CILA_CONTROL0_0

CSI-A Phy and CIL Control

Offset: 21bh | Read/Write: R/W | Reset: 0b0000xx0x0010

Bit	Reset	Description
11:8	0x0	CILA_CLK_SETTLE: settle time for clk lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many csicil clock cycles (72 MHz lp clock cycles) to wait after LP00. HW 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), . . 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 (72 MHz lp clock cycles) to wait, after LP00, before starting to look at the data.

26.11.19 CSI_PHY_CILB_CONTROL0_0

CSI-B Phy and CIL Control

Offset: 21ch | Read/Write: R/W | Reset: 0b0000xx0x0010

Bit	Reset	Description
11:8	0x0	CILB_CLK_SETTLE: settle time for clk lane when moving from LP to HS (LP11->LP01->LP00), this setting determines how many csicil clock cycles (72 MHz lp clock cycles) to wait after LP00. HW 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), . . 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 (72 MHz lp clock cycles) to wait, after LP00, before starting to look at the data.

26.11.20 CSI_CSI_PIXEL_PARSER_STATUS_0

Pixel Parser Status

These status bits are cleared to zero when its bit position is written with one. For example write 0x2 to CSI_PIXEL_PARSER_STATUS will clear only PPA_ILL_WD_CNT.

Offset: 21eh | 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 headers ECC, but can't 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 headers ECC, but can't 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 outputted by the Pixel Parser.
24	X	PPB_EXTRA_SF: Set when CSI-PPB receives a SF when it is expecting an EF. This happens when EF of the frame gets corrupted before arriving CSI. CSI-PPB will insert a fake EF and the 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 doesn't 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, or a CSI RTL bug.
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 a 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 PPB.
17	X	PPB_ILL_WD_CNT: Illegal Word Count, set when a line with a word count that doesn't 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 headers 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 headers ECC, but can't be corrected. The packet will be discarded. a multi bit error. This error will be detected by the headers ECC, but can't 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 can't be corrected. The packet will be discarded.

Bit	Reset	Description
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 outputted by the Pixel Parser.
8	X	PPA_EXTRA_SF: Set when CSI-PPA receives a SF when it is expecting an EF. This happens when EF of the frame gets corrupted before arriving CSI. CSI-PPA will insert a fake EF and the 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 doesn't 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, or a CSI RTL bug.
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 a 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 doesn't 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.

26.11.21 CSI_CSI_CIL_STATUS_0

CSI Control and Interface Logic Status

These status bits are cleared to zero when its bit position is written with one. For example write 0x2 to CSI_CIL_STATUS will clear only CILA_SOT_MB_ERR.

Offset: 21fh | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

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

Bit	Reset	Description
		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 packets 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 packets start of transmission bytes. The packet will be sent to CSI-B for processing.
15	X	MIPI_AUTO_CAL_DONE: MIPI Auto Calibrate done, set when the auto calibrate sequence for MIPI pad bricks is done.
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 packets 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 packets Start of Transmission bytes. The packet will be sent to the CSI-A for processing.

26.11.22 CSI_CSI_PIXEL_PARSER_INTERRUPT_MASK_0

CSI Pixel Parser Interrupt Mask

Offset: 220h | 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 : Don't 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 : Don't 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 : Don't 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 : Don't generate an interrupt when PPB_INTERFRAME_LINE is set. 1 = ENABLED

Bit	Reset	Description
24	0x0	PPB_EXTRA_SF_INT_MASK: Interrupt Mask for PPB_EXTRA_SF. Generate an interrupt when PPB_EXTRA_SF is set. 0 = DISABLED : Don't 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 : Don't 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 : Don't 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 : Don't 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 : Don't 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 : Don't 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 : Don't 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 : Don't 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 : Don't generate an interrupt when PPB_HDR_ERR_COR is set. 1 = ENABLED
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 : Don't 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 : Don't 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 : Don't 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 : Don't 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 : Don't 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.

Bit	Reset	Description
		0 = DISABLED : Don't 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 : Don't 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 : Don't 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 : Don't 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 : Don't 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 : Don't 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 : Don't 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 : Don't generate an interrupt when PPA_HDR_ERR_COR is set. 1 = ENABLED

26.11.23 CSI_CSI_CIL_INTERRUPT_MASK_0

CSI Control and Interface Logic Interrupt Mask

Offset: 221h | Read/Write: R/W | Reset: 0b00000000xxxxxx000000000

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: Don't 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: Don't 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: Don't 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: Don't 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: Don't generate an interrupt when CILB_SYNC_ESC_ERR is set.

Bit	Reset	Description
		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: Don't 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: Don't generate an interrupt when CILB_SOT_SB_ERR is set. 1 = ENABLED
15	0x0	MIPI_AUTO_CAL_DONE_INT_MASK: Interrupt Mask for MIPI_AUTO_CAL_DONE. Generate an interrupt when MIPI_AUTO_CAL_DONE is set. 0 = DISABLED: Don't generate an interrupt when MIPI_AUTO_CAL_DONE 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 : Don't 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: Don't 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 : Don't 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 : Don't 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 : Don't 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 : Don't generate an interrupt when CILA_SYNC_ESC_ERR is set. 1 = ENABLED
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 : Don't 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 : Don't generate an interrupt when CILA_SOT_SB_ERR is set. 1 = ENABLED

26.11.24 CSI_CSI_READONLY_STATUS_0

CSI Read Only Status

This register is used to return CSI read only status.

Offset: 222h | Read/Write: RO | Reset: 0bxx

Bit	Reset	Description
1	X	CSI_PPB_ACTIVE: One only when Pixel Parser B is capturing frame data.

Bit	Reset	Description
0	X	CSI_PPA_ACTIVE: One only when Pixel Parser A is capturing frame data.

26.11.25 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: 223h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
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-B, during the last escape Mode sequence. This command byte can only be assumed to be valid when 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 CILA_ESC_CMD_REC status bit is set.

26.11.26 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: 224h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
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.
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.

26.11.27 CSI_CILA_PAD_CONFIG0_0

CIL-A Pad Configuration 0

Offset: 225h | Read/Write: R/W | Reset: 0b000x0000000x000x000x000x0000111

Bit	Reset	Description
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.
26:24	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.
23:22	0x0	PAD_CILA_LPDNADJ: Driver pull down impedance control 00 -> 130ohm, default 01 -> 110ohm 10 -> 130ohm, same as 00 11 -> 150ohm
21:20	0x0	PAD_CILA_LPUPADJ: Driver pull up impedance control 00 -> 130ohm, default 01 ->

Bit	Reset	Description
		110ohm 10 -> 130ohm, same as 00 11 -> 150ohm
18:17	0x0	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 20ps
10:8	0x0	PAD_CILA_INADJ0: bit 0 input delay trimmer, each tap delays 20ps
6:4	0x0	PAD_CILA_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILA_PREEMP_EN: HS driver preemphasis enable,1= preemphasis enabled
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

26.11.28 CSI_CILA_PAD_CONFIG1_0

CIL-A Pad Configuration 4

Offset: 226h | 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
15:0	RW	0x0	PAD_CILA_SPARE: Spare bits for CILA Config PAD_CILA_SPARE[15] is being used to disable the CSI-A RTL code that blocks fifo push that are past the end of the line packet. 0: disabled, 1: push blocking enabled

26.11.29 CSI_CILB_PAD_CONFIG0_0

CIL-B Pad Configuration 0

Offset: 227h | Read/Write: R/W | Reset: 0b000x0000000xxx0x000x000x0000111

Bit	Reset	Description
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.
26:24	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.
23:22	0x0	PAD_CILB_LPDNADJ: Driver pull down impedance control 00 -> 130ohm, default 01 -> 110ohm 10 -> 130ohm, same as 00 11 -> 150ohm
21:20	0x0	PAD_CILB_LPUPADJ: Driver pull up impedance control 00 -> 130ohm, default 01 -> 110ohm 10 -> 130ohm, same as 00 11 -> 150ohm
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 20ps
10:8	0x0	PAD_CILB_INADJ0: bit 0 input delay trimmer, each tap delays 20ps
6:4	0x0	PAD_CILB_INADJCLK: Clock bit input delay trimmer, each tap delays 20ps
3	0x0	PAD_CILB_PREEMP_EN: HS driver preemphasis enable,1= preemphasis enabled
2	0x1	PAD_CILB_PDIO_CLK: Power down for clock bit, including drivers, receivers and contention detectors

Bit	Reset	Description
1:0	0x3	PAD_CILB_PDIO: Power down for each data bit, including drivers, receivers and contention detectors

26.11.30 CSI_CILB_PAD_CONFIG1_0

CIL-B Pad Configuration 4

Offset: 228h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxx0000000000000000

Bit	R/W	Reset	Description
31:16	RO	X	PAD_CILB_SPARE_RO: Spare Read only bits for CILB Config
15:0	RW	0x0	PAD_CILB_SPARE: Spare bits for CILB Config PAD_CILB_SPARE[15] is being used to disable the CSI-B RTL code that blocks fifo push that are past the end of the line packet. 0: disabled, 1: push blocking enabled

26.11.31 CSI_CIL_PAD_CONFIG0_0

CIL Pad Configuration 0

Offset: 229h | Read/Write: R/W | Reset: 0b00000000x000xx10

Bit	Reset	Description
15:8	0x0	PAD_CIL_SPARE: Spare bit for CIL BIAS Config PAD_CIL_SPARE[7] is used is being used to flush VI's Y-FIFO when it is being use as a stream source for one of the Pixel Parsers. Setting PAD_CIL_SPARE[7] to 1 will hold vi2csi_host_stall low. Which will force 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: 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_CIL_PDVREG: Power down voltage regulator, 1=power down
0	0x0	PAD_CIL_VBYPASS: Bypass bang gap voltage reference

26.11.32 CSI_CILA_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-A MIPI pads

Offset: 22ah | Read/Write: R/W | Reset: 0b00101010xx100000xxx00000xxx00000

Bit	R/W	Reset	Description
31	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 set in order to latch in the over ride value
30	RW	0x0	MIPI_CAL_OVERIDEA: When 0 (normal operation), use the above registers 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.
29:26	RW	0xa	MIPI_CAL_NOISE_FLT: The DRIVRY & TERMRY signals coming from MIPI Pads are utilized by Calibration state machine for PAD Calibration. The drivry/termry comes from a noisy analog source and it could have some glitches. The filter in calibsm is sensitive to these noises. If the calibration done status does not show up, we can change the sensitivity of the filter through these bits. Ideally this has to be programmed in a range from 10 to 15. For the case 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: Set to 00 when calibration step should be 0.1 us Set to 01 when calibration step should be 0.5 us Set to 10 when

Bit	R/W	Reset	Description
			calibration step should be 1.0 us Set to 11 when calibration step should be 1.5 us this will keep the mipi bias cal step between 0.1-1.5 usec Default set for 1.0 us calibration step.
21	RW	0x1	MIPI_CAL_SELA: Select the CSIA PADS for auto calibration.
20:16	RW	0x0	MIPI_CAL_HSPDOSA: 2's complement offset for HSPDADJ going to channel A
12:8	RW	0x0	MIPI_CAL_HSPUOSA: 2's complement offset for HSPUADJ going to channel A
4:0	RW	0x0	MIPI_CAL_TERMOSA: 2's complement offset for TERMADJ going to channel A

26.11.33 CSI_CILB_MIPI_CAL_CONFIG_0

Calibration Settings for CIL-B MIPI pads

Offset: 22bh | Read/Write: R/W | Reset: 0b0xxxxxxxx100000xxx00000xxx00000

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDEB: When 0 (normal operation), use the above registers 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 Bit 31 of CILA_MIPI_CAL_CONFIG (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

26.11.34 CSI_CIL_MIPI_CAL_STATUS_0

CIL MIPI Calibrate Status

Offset: 22ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

Bit	Reset	Description
15:12	X	MIPI_CAL_DRIV_DN_ADJ: Driver Pull down calibration code generated by MIPI auto Calibrate. Valid only after auto calibrate sequence has completed (MIPI_CAL_ACTIVE == 0).
11:8	X	MIPI_CAL_DRIV_UP_ADJ: Driver 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 auto calibrate sequence has completed (MIPI_CAL_ACTIVE == 0).
0	X	MIPI_CAL_ACTIVE: One when auto calibrate is active.

26.11.35 CSI_DSI_MIPI_CAL_CONFIG_0

Calibration Settings for DSI MIPI pad

Offset: 234h | Read/Write: R/W | Reset: 0b0xxxxxxxx100000xxx00000xxx00000

Bit	Reset	Description
30	0x0	MIPI_CAL_OVERIDED: When 0 (normal operation), use the above registers 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 Bit 31 of

Bit	Reset	Description
		CILA_MIPI_CAL_CONFIG (MIPI_CAL_STARTCAL) starts the Calibration State machine.
21	0x1	MIPI_CAL_SELD: Select the DSI PADS for auto calibration.
20:16	0x0	MIPI_CAL_HSPDOSD: 2's complement offset for HSPDADJ
12:8	0x0	MIPI_CAL_HSPUOSD: 2's complement offset for HSPUADJ
4:0	0x0	MIPI_CAL_TERMOSD: 2's complement offset for TERMADJ

26.11.36 CSI_MIPIBIAS_PAD_CONFIG0_0

MIPI Bias Pad Configuration Register

Offset: 235h | Read/Write: R/W | Reset: 0b000xxxxx000xxxxx000xxxxx000

Bit	Reset	Description
26:24	0x0	PAD_TEST_SEL: Controls which signal to be routed to TEST_OUT 000=VAUXP 001=RUP 010=RDN 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

26.11.37 CSI_CSI_CILA_STATUS_0

CSI-CILA Control and Interface Logic Status

These status bits are cleared to zero when its bit position is written with one. For example write 0x2 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 236h | Read/Write: R/O | Reset: 0bxxxxxxxxxxxxxxxxxxxx

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 packets 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 packets 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 packets 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 clock lane

26.11.38 CSI_CSI_CILB_STATUS_0

CSI-CILB Control and Interface Logic Status

These status bits are cleared to zero when its bit position is written with one. For example write 0x2 to CSI_CIL_STATUS will clear only SOT_MB_ERR.

Offset: 237h | Read/Write: R/O | Reset: 0bxxxxxxxxxxxxxxxxxx

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 packets 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-A detects a single bit error in one of the packets 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 packets 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 packets Start of Transmission bytes on data lane-0. The packet will be sent to the CSI-B for processing.
0	X	CILB_CLK_LANE_CTRL_ERR: Control Error, set when CIL-B detects incorrect line state sequence on clock lane

[THIS PAGE INTENTIONALLY LEFT BLANK]

27.0 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 it.

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. The register interface is documented to aid with understanding those drivers.

27.1 VI Registers

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

27.1.1 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 | 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

27.1.2 VI_INTSTATUS_0

Offset: 0x21 | Read/Write: RO | Reset: 0x00000000 (0b0)

Bit	Reset	Description
0	0x0	CTXSW_INT: Context switch interrupt status (clear on write)

27.1.3 VI_VI_INPUT_CONTROL_0

VI Input Control

For Parallel VIP input, limitation for vsync and hsync has to be followed to avoid ISP hang:

Software must always program parallel cameras (including the VIP pattern generator) in a way that avoids simultaneous HSYNC and VSYNC active edges.

Offset: 0x22 | Read/Write: R/W | Reset: 0x00000000 (0b00000000xxxxxxxxxx0000000000000)

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: Interlaced video Field Detection (effective if VIDEO_SOURCE is VIP) 0= Disabled (top field only) 1= Enabled When H/V syncs are decoded per ITU-R BT.656 standard, odd/even field is detected from the control bytes. When H/V syncs come from VHS/VVS pins (YUV422), odd/even field is detected from the position of VVS active edge with respect to VHS active pulse. This bit should be disabled for non-interlaced source or when H/V syncs are generated internally. If VIDEO_SOURCE is HOST, field information is always specified by host. 0 = DISABLED 1 = ENABLED
26:25	0x0	SYNC_FORMAT: Horizontal and Vertical Sync Format (effective if VIDEO_SOURCE is VIP) 00= horizontal sync comes from VHS pin and vertical sync comes from VVS pin consistent with standard YUV422 data format. In this case, VHS_Input_Control and VVS_Input_Control must be enabled. 01= horizontal and vertical syncs are decoded from the received video data bytes as specified in ITU-R BT.656 (CCIR656) standard. 10= horizontal and vertical syncs are generated internally and they are output on VHS and VVS pins if VHS and VVS are in output mode. 0 = YUV422 1 = ITU656 2 = INTHVS
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)

Bit	Reset	Description
		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 input source is VI Port and format is YUV422/ITU-R BT.656 or when input source is host and 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 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: Input Port Data Format (effective if input source is VI Port) 0000= YUV422 or ITU-R BT.656 0001= Reserved 1 0010= Bayer Pattern, enables ISP 0011= Reserved 2 0100= Pattern A, written directly to memory 0101= Reserved 3 0110= Pattern C, written directly to memory 0111= Pattern C, do not remove the 0xFF, 0x02 1000= Pattern D, ISDB-T input 1001= YUV420NP, written directly to memory as YUV420P 1010= RGB565, written directly to EPP 1011= RGB888, written directly to EPP 1100= RGB444, written directly to EPP 1101= CSI, written directly to CSI For YUV420NP no cropping will be done. For RGB565, RGB888, RGB444 written to EPP all cropping will be done in the EPP. 0 = YUV422 1 = RESERVED_1 2 = BAYER 3 = RESERVED_2 4 = PATTERN_A 5 = RESERVED_3 6 = PATTERN_C 7 = PATTERN_C_RAW 8 = PATTERN_D 9 = YUV420 10 = RGB565 11 = RGB888 12 = RGB444 13 = CSI

Bit	Reset	Description
1	0x0	VIP_INPUT_ENABLE: VIP Input Enable. This bit turns on clocks for VIP input logic. This bit has to be enabled before CAMERA_CONTROL's VIP_ENABLE bit for any VIP logic to start! 0 = DISABLED 1 = ENABLED
0	0x0	HOST_INPUT_ENABLE: Host Input Enable 0 = DISABLED 1 = ENABLED

27.1.4 VI_VI_CORE_CONTROL_0

VI Core Control and Output to EPP/ISP

Offset: 0x23 | 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 000= Parallel Video Input Port data 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 0 = VIP 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
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

Bit	R/W	Reset	Description
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 00= Parallel Video Input Port data 01= Host I/F data 10= ISP data, from 444 to 422 converter 11= reserved 0 = VIP 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 11 = Averaged, average the odd and even UVs. (Not Implemented) 0 = COSITED_EVEN 1 = COSITED_ODD 2 = NONCOSITED 3 = AVERAGED

Bit	R/W	Reset	Description
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 down-scaling 010= YUV444 stream before down-scaling 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 down-scaling 100= RGB565,RGB444,RGB888 from VIP, no LPF or downscaling 101= RGB565,RGB888 from Host 110= CSI_PPA 111= CSI_PPB 0 = DISABLED 1 = YUV444POST 2 = YUV444PRE 3 = YUV444ISP 4 = RGB 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= Parallel Video Input Port data 10= Host I/F data 11= stereo compositor 0 = DISABLED 1 = VIP 2 = HOST 3 = STEREO

27.1.5 VI_VI_FIRST_OUTPUT_CONTROL_0

VI Output Control of YUV/RGB and YUV420P

Offset: 0x24 | Read/Write: R/W | Reset: 0x00000000 (0b0000000xxxxxxx0xxxxx000)

Bit	Reset	Description
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
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

Bit	Reset	Description
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

27.1.6 VI_VI_SECOND_OUTPUT_CONTROL_0

VI Second Output Control of YUV422NP and RGB

Offset: 0x25 | Read/Write: R/W | Reset: 0x00000000 (0b000000xxxxxxxxxxxx0000)

Bit	Reset	Description
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
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

Bit	Reset	Description
		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= VIP Bayer direct to memory as a 16-bit value {6'b0, VIP_pad[9:0]} 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=VIP_BAYER_DIRECT: Bayer data is written unmodified to memory as a 16-bit quantity. Bit 0 of incoming data is placed in bit 0 of the 16-bit memory location. Upper bits are padded with 0s. 1011=YUV422 stream from the ALT mux 0 = RGB16 1 = RGB16D 2 = RGB24 3 = YUV422POST 4 = YUV422PRE 5 = JPEG_STREAM 6 = VIP_BAYER 7 = CSI_PPA_BAYER 8 = CSI_PPB_BAYER 9 = VIP_BAYER_DIRECT 10 = YUV422ALT

27.1.7 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.8 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 | Read/Write: R/W | Reset: 0x00000000 (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).

Bit	Reset	Description
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.

27.1.9 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.10 VI_VIP_H_ACTIVE_0

VI Horizontal Active

This register defines the horizontal active area of the input video source with respect to horizontal sync. (This is for VIP data.)

Offset: 0x29 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	VIP_H_ACTIVE_PERIOD: Horizontal Active Period. This parameter specifies the number of pixels in the horizontal active area. The value is the END of the active region, so PERIOD-START = active area This parameter should be programmed with an even number
12:0	X	VIP_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.

27.1.11 VI_VIP_V_ACTIVE_0

Vertical Active

This register defines the vertical active area of the input video source with respect to vertical sync. (This is for VIP data.)

Offset: 0x2a | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
28:16	X	VIP_V_ACTIVE_PERIOD: Vertical Active Period. This parameter specifies the number of lines in the vertical active area. The value is the END of the active region, so PERIOD-START = active area
12:0	X	VIP_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.

27.1.12 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 | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

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

27.1.13 VI_VI_DMA_SELECT_0

Host DMA select

Offset: 0x2c | Read/Write: R/W | Reset: 0x00000000 (0b00)

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

27.1.14 VI_HOST_DMA_WRITE_BUFFER_0

Host DMA Write Buffer Configuration Registers

Offset: 0x2d | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
27:26	0x0	SOURCE_SEL: Data source selection 00= VIP (backward compatible) 01= CSI_PPA 10= CSI_PPB 0 = VIP 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

27.1.15 VI_HOST_DMA_BASE_ADDRESS_0

Host DMA Write Buffer Configuration Registers

Offset: 0x2e | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	DMA_BASE_ADDR: Base Address

27.1.16 VI_HOST_DMA_WRITE_BUFFER_STATUS_0

Host DMA Write Buffer Status Register

Offset: 0x2f | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
26:0	X	WB_STATUS: Status of the Host DMA write buffer

27.1.17 VI_HOST_DMA_WRITE_PEND_BUFCOUNT_0

Host DMA Write Buffer Pending Buffer Count

Offset: 0x30 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxx)

Bit	Reset	Description
8:0	X	PEND_BUFCOUNT: Pending buffer count of the Host DMA write buffer.

27.1.18 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.19 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.20 VI_VB0_START_ADDRESS_U_0

Video Buffer O Start Address U (linked to First Output)

Offset: 0x33 | Read/Write: R/W | Reset: 0x00000000 (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

27.1.21 VI_VB0_BASE_ADDRESS_U_0

Video Buffer O BASE Address U (linked to First Output)

Offset: 0x34 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.22 VI_VB0_START_ADDRESS_V_0

Video Buffer O Start Address V (linked to First Output)

Offset: 0x35 | Read/Write: R/W | Reset: 0x00000000 (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

27.1.23 VI_VB0_BASE_ADDRESS_V_0

Video Buffer O BASE Address V (linked to First Output)

Offset: 0x36 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.24 VI_VB0_SCRATCH_ADDRESS_UV_0

Video Buffer O Scratch Address UV (linked to First Output)

Offset: 0x37 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.25 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 | Read/Write: R/W | Reset: 0x00000000 (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

27.1.26 VI_VB0_COUNT_FIRST_0

Video Buffer Set 0 Count for First Output

Offset: 0x39 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx)

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.

27.1.27 VI_VB0_SIZE_FIRST_0

Video Buffer Set 0 Size for First Output

Offset: 0x3a | Read/Write: R/W | Reset: 0x00000000 (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

27.1.28 VI_VB0_BUFFER_STRIDE_FIRST_0

Video Buffer Set 0 Buffer Stride

Offset: 0x3b | Read/Write: R/W | Reset: 0x00000000 (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)

27.1.29 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.30 VI_VB0_BASE_ADDRESS_SECOND_0

Video Buffer O Base Address for Second Output

Offset: 0x3d | Read/Write: R/W | Reset: 0x00000000 (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

27.1.31 VI_SECOND_OUTPUT_FRAME_SIZE_0

Width and Height of Second Output Frame

Offset: 0x3e | Read/Write: R/W | Reset: 0x00000000 (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

27.1.32 VI_VB0_COUNT_SECOND_0

Video Buffer Set 0 Count for Second Output

Offset: 0x3f | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxx)

Bit	Reset	Description
7:0	X	VB0_COUNT_2: This specifies the number of buffers in video buffer set 0.

27.1.33 VI_VB0_SIZE_SECOND_0

Video Buffer Set 0 Size for Second Output

Offset: 0x40 | Read/Write: R/W | Reset: 0x00000000 (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).

27.1.34 VI_VB0_BUFFER_STRIDE_SECOND_0

Video Buffer Set 0 Buffer Stride for Second Output

Offset: 0x41 | Read/Write: R/W | Reset: 0x00000000 (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)

27.1.35 VI_H_LPF_CONTROL_0

VI Horizontal Low-Pass Filter (LPF) Control

Offset: 0x42 | Read/Write: R/W | Reset: 0x02400240 (0b0001001000000xxx0001001000000)

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.

27.1.36 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 down-scaling (decimation) is performed without horizontal averaging, the down-scaling factor is specified by input active period and output frame size. Because the VI has two input methods (VIP and HOST) and two memory outputs, there are mux selects to indicate which registers to use in calculating the input and output frame sizes.

If horizontal downscaling is performed with horizontal averaging, the down-scaling 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 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000xxxxxxxxxxxx00xx)

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. Selects between the VIP and HOST input active area widths for the denominator in the downscaling ratio. Uses VIP_H_ACTIVE_PERIOD or 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. 0 = VIP 1 = HOST

27.1.37 VI_V_DOWNSCALE_CONTROL_0

VI Vertical Down-scaling 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. Because the VI has two input methods (VIP and HOST) and two memory outputs, there are mux selects to indicate which registers to use in calculating the input and output frame sizes.

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 | Read/Write: R/W | Reset: 0x00000000 (0b0000000000000xxxxxxxxxxxx00xx)

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

Bit	Reset	Description
0	X	INPUT_V_SIZE_SEL: Input Vertical Size Select Selects between the VIP and HOST input active area heights for the denominator in the downscaling ratio. Uses VIP_V_ACTIVE_PERIOD or 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. 0 = VIP 1 = HOST

27.1.38 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 | Read/Write: R/W | Reset: 0x00000000 (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

27.1.39 VI_CSC_UV_R_0

CSC U & V coefficient for R

Offset: 0x46 | Read/Write: R/W | Reset: 0x00000000 (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

27.1.40 VI_CSC_UV_G_0

CSC U & V coefficient for G

Offset: 0x47 | Read/Write: R/W | Reset: 0x00000000 (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

27.1.41 VI_CSC_UV_B_0

CSC U & V coefficient for B

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

27.1.42 VI_CSC_ALPHA_0

RGB Color Space Converter Alpha value

Offset: 0x49 | Read/Write: R/W | Reset: 0x00000000 (0b00000000)

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.

27.1.43 VI_HOST_VSYNC_0

Valid when INPUT_SOURCE is HOST

Offset: 0x4a | Read/Write: RO | Reset: 0x00000000 (0bx)

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

27.1.44 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 | Read/Write: R/W | Reset: 0x00000000 (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.

Bit	Reset	Description
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

27.1.45 VI_HOST_FIFO_STATUS_0

Host FIFO status

This is not needed if host input video goes through command buffer interface.

Offset: 0x4c | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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.

27.1.46 VI_INTERRUPT_MASK_0

Interrupt Mask

Offset: 0x4d | Read/Write: R/W | Reset: 0x00000000 (0b00000000x00000000000000000000)

Bit	Reset	Description
28	0x0	SECOND_OUTPUT_DROP_MC_DATA_INT_MASK: This bit controls interrupt when VI drops data to the MC. 0 = DISABLED 1 = ENABLED
27	0x0	FIRST_OUTPUT_DROP_MC_DATA_INT_MASK: This bit controls interrupt when VI drops data to the 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

Bit	Reset	Description
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
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

Bit	Reset	Description
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
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

27.1.47 VI_INTERRUPT_TYPE_SELECT_0

Interrupt Type Select

Offset: 0x4e | Read/Write: R/W | Reset: 0x00000000 (0b000000000)

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

Bit	Reset	Description
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
0	0x0	VD8_INT_TYPE: VD8 pin Interrupt Type. This bit controls interrupt VD8 if edge or level type 0 = EDGE 1 = LEVEL

27.1.48 VI_INTERRUPT_POLARITY_SELECT_0

Interrupt Polarity Select

Offset: 0x4f | Read/Write: R/W | Reset: 0x00000000 (0b000000000)

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

Bit	Reset	Description
		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

27.1.49 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 | Read/Write: RO | Reset: 0x00000000 (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 an other 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 get 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 get 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

Bit	Reset	Description
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
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 is forced to 0 if FIELD_DETECT is DISABLED when VIDEO_SOURCE is VIP. 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

Bit	Reset	Description
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
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

Bit	Reset	Description
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 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

27.1.50 VI_VIP_INPUT_STATUS_0

Video Input Port status

Offset: 0x51 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:16	X	FRAME_COUNT: The number of frames received (VSYNCs). Any write to this register clears this.
15:0	X	LINE_COUNT: The number of lines received (HSYNCs)

27.1.51 VI_VIDEO_BUFFER_STATUS_0

Interrupt Enable

Offset: 0x52 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

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

27.1.52 VI_SYNC_OUTPUT_0

VI H and V Sync Output control

This register controls VHS and VVS output when H/V syncs are generated internally in the VI module (VIDEO_SOURCE is VIP and SYNC_FORMAT is INTHVS).

The generated VHS and VVS signal can be sent to external video source device and used to synchronize the video data transfer from the video source to the VI module. VHS and VVS pin should be configured in output mode to output the internally generated H/V syncs.

Also in this case, the internally generate H/V syncs can be used by the VI module as horizontal and vertical reference signals for the incoming video data.

Offset: 0x53 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:19	X	VVS_OUTPUT_PERIOD: This specifies VVS output pulse period in term of number of VHS cycles. Programmed value is actual value - 1 so valid value ranges from 2 to 4096.
18:16	X	VVS_OUTPUT_WIDTH: This specifies VVS output pulse width in term of number of VHS cycles. Programmed value is actual value - 1 so valid value ranges from 1 to 8.
15:3	X	VHS_OUTPUT_PERIOD: This specifies VHS output pulse period in term of number of VI clock cycles. Programmed value is actual value - 1 so valid value ranges from 32 to 8192.
2:0	X	VHS_OUTPUT_WIDTH: This specifies VHS output pulse width in term of number of VI clock cycles. Programmed value is actual value - 1 so valid value ranges from 1 to 8.

27.1.53 VI_VVS_OUTPUT_DELAY_0

VI V Sync Output Delay

Offset: 0x54 | Read/Write: R/W | Reset: 0x00000000 (0bxxxx)

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.

27.1.54 VI_PWM_CONTROL_0

VI Pulse Width Modulation Control

VI Pulse Width Modulation signal generation

PWM signal generation logic can generate up to 128 pulses per line internally and the PWM pulse select registers determines which of the 128 pulses will be output. Any of the 128 internally generated pulse can be independently selected as output if they occur within one line time.

PWM signal can be output on the VGP6 pin if VGP6 output is enabled and the output select is set to PWM.

The PWM will be triggered by the first vsync after the PWM_ENABLE bit has been set.

Offset: 0x55 | Read/Write: R/W | Reset: 0x00000000 (0b00000000xx00xxxx00000000xxx0xxx0)

Bit	Reset	Description
31:24	0x0	PWM_COUNTER: PWM Counter 8-bit value used when PWM_MODE is set to COUNTER to determine how many times the PWM will cycle through the 128 cycles before stopping.
21:20	0x0	PWM_MODE: PWM Mode Continuous - after PWM is turned on, continue through the PWM's 128 cycles repeatedly until the PWM is turned off. Single - after PWM is turned on, cycle once through the 128 cycles and stop. Counter - after PWM is turned on, cycle through the 128 cycles PWM_COUNTER number of times then stop. 0 = CONTINUOUS 1 = SINGLE 2 = COUNTER
15:12	0x0	PWM_LOW_PULSE: PWM Low Pulse (1 to 16)
11:8	0x0	PWM_HIGH_PULSE: PWM High Pulse (1 to 16)

Bit	Reset	Description
4	0x0	PWM_DIRECTION: PWM Direction 0= Incrementing 1= Decrementing 0 = INCR 1 = DECR
0	0x0	PWM_ENABLE: PWM Enable 0 = DISABLED 1 = ENABLED

27.1.55 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 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_A: PWM Select bits 31 to 0

27.1.56 VI_PWM_SELECT_PULSE_B_0

PWM Pulse Select B

Offset: 0x57 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_B: PWM Select bits 63 to 32

27.1.57 VI_PWM_SELECT_PULSE_C_0

PWM Pulse Select C

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

Bit	Reset	Description
31:0	X	PWM_SELECT_C: PWM Select bits 95 to 64

27.1.58 VI_PWM_SELECT_PULSE_D_0

PWM Pulse Select D

Offset: 0x59 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	PWM_SELECT_D: PWM Select bits 127 to 96

27.1.59 VI_VI_DATA_INPUT_CONTROL_0

VI Input Mask

Offset: 0x5a | Read/Write: R/W | Reset: 0x00000fff (0b111111111111)

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

27.1.60 VI_PIN_INPUT_ENABLE_0

VI pins Input Enable

Offset: 0x5b | Read/Write: R/W | Reset: 0x00000000 (0b000000x00x000000000000)

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
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

Bit	Reset	Description
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

27.1.61 VI_PIN_OUTPUT_ENABLE_0

VI pins Output Enable

Offset: 0x5c | Read/Write: R/W | Reset: 0x00000000 (0b000000x000000000000000)

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
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

Bit	Reset	Description
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

27.1.62 VI_PIN_INVERSION_0

VI Pins Input/Output Inversion 0 reserved

Offset: 0x5d | Read/Write: R/W | Reset: 0x00000000 (0b000xxxxxxxxxxxx00)

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

Bit	Reset	Description
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

27.1.63 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 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx)

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
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

Bit	Reset	Description
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

27.1.64 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 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxx)

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)

Bit	Reset	Description
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)
0	X	VD0_OUTPUT_DATA: VD0 pin Output Data (effective if VD0_OUTPUT_ENABLE is ENABLED and VD0_OUTPUT_SELECT is DATA)

27.1.65 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 | Read/Write: R/W | Reset: 0x00000000 (0b000000x000000000000000)

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
0	0x0	PIN_OUTPUT_SELECT_vd0: Pin Output Select VD0

27.1.66 VI_RAISE_VIP_BUFFER_FIRST_OUTPUT_0

Raise vector at buffer end

Raise vectors are received from host. If host is the input source, the host will send a raise vector at the end of a line, and the VI returns it when that has been written to memory.

A raise written when decimation or averaging is selected in the VI is not supported.

If the Video Input Port is the input source, the host should program raise vectors to either raise at buffer end or at frame end.

Since there are 2 memory outputs for VI, there are two separate raise vectors for buffer/frame.

Offset: 0x61 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_BUFFER_1_CHANNEL
4:0	RW	X	RAISE_BUFFER_1_VECTOR

27.1.67 VI_RAISE_VIP_FRAME_FIRST_OUTPUT_0

Raise vector at frame end

Offset: 0x62 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_FRAME_1_CHANNEL
4:0	RW	X	RAISE_FRAME_1_VECTOR

27.1.68 VI_RAISE_VIP_BUFFER_SECOND_OUTPUT_0

Raise vector at buffer end

Offset: 0x63 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_BUFFER_2_CHANNEL
4:0	RW	X	RAISE_BUFFER_2_VECTOR

27.1.69 VI_RAISE_VIP_FRAME_SECOND_OUTPUT_0

Raise vector at frame end

Offset: 0x64 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_FRAME_2_CHANNEL
4:0	RW	X	RAISE_FRAME_2_VECTOR

27.1.70 VI_RAISE_HOST_FIRST_OUTPUT_0

Raise Vector When from Host

Offset: 0x65 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_HOST_1_CHANNEL
4:0	RW	X	RAISE_HOST_1_VECTOR

27.1.71 VI_RAISE_HOST_SECOND_OUTPUT_0

Raise Vector When from Host

Offset: 0x66 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_HOST_2_CHANNEL

Bit	R/W	Reset	Description
4:0	RW	X	RAISE_HOST_2_VECTOR

27.1.72 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 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_EPP_CHANNEL
4:0	RW	X	RAISE_EPP_VECTOR

27.1.73 VI_CAMERA_CONTROL_0

VI camera control bits

For Parallel VIP input, limitation for vsync and hsync has to be followed to avoid ISP hang for AP15:

Software must always program parallel cameras (including the VIP pattern generator) in a way that avoids simultaneous HSYNC and VSYNC active edges.

Offset: 0x68 | Read/Write: R/W | Reset: 0x00000000 (0b000)

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: VI camera input module Enable 0= Ignored - use the STOP_CAPTURE to turn off the capturing 1= Enabled Write a 1'b1 to this register to enable the camera interface to start capturing data. 0 = DISABLED 1 = ENABLED

27.1.74 VI_VI_ENABLE_0

VI Enables

Offset: 0x69 | Read/Write: R/W | Reset: 0x00000001 (0b01)

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

27.1.75 VI_VI_ENABLE_2_0

VI Enables second output

Offset: 0x6a | Read/Write: R/W | Reset: 0x00000001 (0b01)

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

27.1.76 VI_VI_RAISE_0

VI Enables second output

Offset: 0x6b | Read/Write: R/W | Reset: 0x00000000 (0b0)

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

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

Bit	Reset	Description
31:0	X	Y_FIFO_DATA

27.1.78 VI_U_FIFO_WRITE_0

YUV 4:2:0 Planar U-FIFO

Offset: 0x6d | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	U_FIFO_DATA

27.1.79 VI_V_FIFO_WRITE_0

YUV 4:2:0 Planar V-FIFO

Offset: 0x6e | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Bit	Reset	Description
31:0	X	V_FIFO_DATA

27.1.80 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 2nd-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 | Read/Write: R/W | Reset: 0x00000000 (0b00xxxxxxxxxxxx0000)

Bit	Reset	Description
17	0x0	VI_RCLK_OVERRIDE
16	0x0	VI_WCLK_OVERRIDE
3	DISABLE	VI_MCCIF_RDCL_RDFAST: 0 = DISABLE 1 = ENABLE
2	DISABLE	VI_MCCIF_WRMC_CLLE2X: 0 = DISABLE 1 = ENABLE
1	DISABLE	VI_MCCIF_RDMC_RDFAST: 0 = DISABLE 1 = ENABLE
0	DISABLE	VI_MCCIF_WRCL_MCLE2X: 0 = DISABLE 1 = ENABLE

27.1.81 VI_TIMEOUT_WCOAL_VI_0

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 | 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

27.1.82 VI_MCCIF_VIRUV_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: 0x71 | Read/Write: R/W | Reset: 0x00000000 (0b000000xxxxxxxxxxxx0000)

Bit	Reset	Description
21:16	0x0	CBR_VIRUV2MC_HPTM
3:0	0x0	CBR_VIRUV2MC_HPTH

27.1.83 VI_MCCIF_VIWSB_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: 0x72 | Read/Write: R/W | Reset: 0x00000000 (0b0000000)

Bit	Reset	Description
6:0	0x0	CBW_VIWSB2MC_HPTH

27.1.84 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 | Read/Write: R/W | Reset: 0x00000000 (0b0000000)

Bit	Reset	Description
6:0	0x0	CBW_VIWU2MC_HPTH

27.1.85 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 | Read/Write: R/W | Reset: 0x00000000 (0b0000000)

Bit	Reset	Description
6:0	0x0	CBW_VIWV2MC_HPTH

27.1.86 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 | Read/Write: R/W | Reset: 0x00000000 (0b0000000)

Bit	Reset	Description
6:0	0x0	CBW_VIWY2MC_HPTH

27.1.87 VI_CSI_PPA_RAISE_FRAME_START_0

CSI Pixel Parser A Raise

CSI Raise vectors

Offset: 0x76 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

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.

27.1.88 VI_CSI_PPA_RAISE_FRAME_END_0

CSI Pixel Parser A Raise

Offset: 0x77 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

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.

27.1.89 VI_CSI_PPB_RAISE_FRAME_START_0

CSI Pixel Parser B Raise

Offset: 0x78 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

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.

27.1.90 VI_CSI_PPB_RAISE_FRAME_END_0

CSI Pixel Parser B Raise

Offset: 0x79 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxx)

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.

27.1.91 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.92 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.93 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.94 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.95 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 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx)

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).

27.1.96 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 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxx)

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).

27.1.97 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 | Read/Write: R/W | Reset: 0x00000000 (0b000000000000)

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

Bit	Reset	Description
0	0x0	VIP_INPUT_1: 0 = NOT_USED 1 = USED

27.1.98 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 | Read/Write: R/W | Reset: 0x00000000 (0b000000000000)

Bit	Reset	Description
11	0x0	HOST_DMA_BUFFER_OUTPUT_2: 0 = NOT_USED 1 = USED
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: 0 = NOT_USED 1 = USED

27.1.99 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 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_CHANNEL_STREAM_1
4:0	RW	X	RAISE_VECTOR_STREAM_1

27.1.100 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 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxxxxxx)

Bit	R/W	Reset	Description
19:16	WO	X	RAISE_CHANNEL_STREAM_2
4:0	RW	X	RAISE_VECTOR_STREAM_2

27.1.101 VI_TS_MODE_0

ISDB-T Mode Selection Register

ISDB-T tuner mode register settuner/demodulator mode.

Offset: 0x84 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxx)

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

27.1.102 VI_TS_CONTROL_0

ISDB-T Mode Control Register

Offset: 0x85 | Read/Write: R/W | Reset: 0x00000008 (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
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

27.1.103 VI_TS_PACKET_COUNT_0

ISDB-T Packet Count Register

Offset: 0x86 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx)

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.

27.1.104 VI_TS_ERROR_COUNT_0

ISDB-T Error Count Register

Offset: 0x87 | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxxx)

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.

27.1.105 VI_TS_CPU_FLOW_CTL_0

ISDB-T CPU Flow Control Register

Offset: 0x88 | Read/Write: RO | Reset: 0x00000000 (0bx)

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.

27.1.106 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 | Read/Write: R/W | Reset: 0x00000000 (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

27.1.107 VI_VB0_CHROMA_LINE_STRIDE_FIRST_0

Video Buffer Set 0 Chroma Line Stride for First Output of Planar YUV formats

Offset: 0x8a | Read/Write: R/W | Reset: 0x00000000 (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).

27.1.108 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 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxx)

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.

27.1.109 VI_BUFFER_RELEASE_OUTPUT1_0

Writes to this register will decrease BUFFER_COUNTER by 1.

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

Bit	Reset	Description
0	X	BUFFER_RELEASE_OUTPUT1

27.1.110 VI_BUFFER_RELEASE_OUTPUT2_0

Offset: 0x8d | Read/Write: R/W | Reset: 0x00000000 (0bx)

Bit	Reset	Description
0	X	BUFFER_RELEASE_OUTPUT2

27.1.111 VI_DEBUG_FLOW_CONTROL_COUNTER_OUTPUT1_0

Debug register

Offset: 0x8e | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxx)

Bit	Reset	Description
7:0	X	BUFFER_COUNT_OUTPUT1

27.1.112 VI_DEBUG_FLOW_CONTROL_COUNTER_OUTPUT2_0

Offset: 0x8f | Read/Write: RO | Reset: 0x00000000 (0bxxxxxxxx)

Bit	Reset	Description
7:0	X	BUFFER_COUNT_OUTPUT2

27.1.113 VI_TERMINATE_BW_FIRST_0

Writes to this register will terminate MC on BW operation in the FIRST output.

Offset: 0x90 | Read/Write: R/W | Reset: 0x00000000 (0bx)

Bit	Reset	Description
0	X	TERMINATE_FIRST_BW

27.1.114 VI_TERMINATE_BW_SECOND_0

Writes to this register will terminate MC on BW operation in the SECOND output.

Offset: 0x91 | Read/Write: R/W | Reset: 0x00000000 (0bx)

Bit	Reset	Description
0	X	TERMINATE_SECOND_BW

27.1.115 VI_VB0_FIRST_BUFFER_ADDR_MODE_0

Memory Controller Tiling definitions

Offset: 0x92 | Read/Write: R/W | Reset: 0x00000000 (0b0xxxxxxx0)

Bit	Reset	Description
8	0x0	UV1_TILE_MODE: 0 = LINEAR 1 = TILED
0	0x0	Y1_TILE_MODE: 0 = LINEAR 1 = TILED

27.1.116 VI_VB0_SECOND_BUFFER_ADDR_MODE_0

Offset: 0x93 | Read/Write: R/W | Reset: 0x00000000 (0b0)

Bit	Reset	Description
0	0x0	Y2_TILE_MODE: 0 = LINEAR 1 = TILED

27.1.117 VI_RESERVE_0_0

Reserved register

Bits [13:0] are reserved for the VIP Pattern Generator (Pattern Width).

VIP Pattern Generator

The VIP Pattern Generator, when enabled, overrides the inputs from the attached camera with internally generated pattern data, HSYNCs, and VSYNCs. The purpose of the pattern generator is to facilitate regression testing of the VI driver and hardware without constraining the board level design.

The pattern generator logic runs on the pd2vi_clock domain. See the CLK_RST_CONTROLLER_CLK_SOURCE_VI_0 register for how to enable a loopback from the vi_sensor clock to the pd2vi_clock.

The user must program the pattern width, pattern height, and Bayer select registers prior to enabling the pattern generator. It is illegal to change the values of those registers without first disabling the pattern generator.

The pattern generator has no concept of blanking time. The width and the height of the pattern should correspond to the full hblank+hactive and vblank+vactive.

Offset: 0x94 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:12	X	nc_RESERVE_0_3
11:8	X	nc_RESERVE_0_2: there are 2 clocks per pixel for YUV422)

Bit	Reset	Description
7:4	X	nc_RESERVE_0_1: pattern width in clocks.
3:0	X	nc_RESERVE_0_0: Program to *one less* than the desired value

27.1.118 VI_RESERVE_1_0

Reserved register

Bits [13:0] are reserved for the VIP Pattern Generator (Pattern Height)

Offset: 0x95 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

Bit	Reset	Description
15:12	X	nc_RESERVE_1_3
11:8	X	nc_RESERVE_1_2
7:4	X	nc_RESERVE_1_1: Pattern height in lines
3:0	X	nc_RESERVE_1_0: Program to one less than the desired value

27.1.119 VI_RESERVE_2_0

Reserved register

Bit 0 is reserved for VIP Pattern Gen Enable. Bit 1 is reserved for VIP Pattern Generator Bayer Select.

Offset: 0x96 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxx0000)

Bit	Reset	Description
15:12	X	nc_RESERVE_2_3
11:8	X	nc_RESERVE_2_2
7:4	X	nc_RESERVE_2_1
3:0	0x0	nc_RESERVE_2_0: 1: BAYER pattern 0: YUV pattern

27.1.120 VI_RESERVE_3_0

Reserved register

Offset: 0x97 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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

27.1.121 VI_RESERVE_4_0

Reserved register

Offset: 0x98 | Read/Write: R/W | Reset: 0x00000000 (0bxxxxxxxxxxxxxxxx)

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

Memory Client Interface Hysteresis Registers

27.1.122 VI_MCCIF_VIRUV_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: 0x99 | Read/Write: R/W | Reset: 0xcf04ff06 (0b11001111000001001111111100000110)

Bit	Reset	Description
31	ENABLE	CBR_ VIRUV2MC_HYST_EN: 1 = ENABLE 0 = DISABLE
30:28	0x4	CBR_ VIRUV2MC_HYST_REQ_TH
27:24	0xf	CBR_VIRUV2MC_HYST_TM
23:16	0x4	CBR_VIRUV2MC_DHYST_TH
15:8	0xff	CBR_VIRUV2MC_DHYST_TM
7:0	0x6	CBR_VIRUV2MC_HYST_REQ_TM

27.1.123 VI_MCCIF_VIWSB_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: 0x9a | 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
11:0	0x1ff	CBW_VIWSB2MC_HYST_REQ_TM

27.1.124 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 | Read/Write: R/W | Reset: 0xc00001ff (0b1100xxxxxxxxxxxxxxx000111111111)

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

27.1.125 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 | Read/Write: R/W | Reset: 0xc00001ff (0b1100xxxxxxxxxxxxxxx000111111111)

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

27.1.126 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 | 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

27.1.127 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 | Read/Write: RO | Reset: 0x00000000 (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

27.1.128 VI_FIELD_STATUS_SHADOW2_0

Offset: 0x9f | Read/Write: RO | Reset: 0x00000000 (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

27.1.129 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.130 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.131 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.132 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 | Read/Write: R/W | Reset: 0x00000000 (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.

27.1.133 VI_VI_STEREO_CONTROL_0

This register controls the aspects of stereo capture.

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

27.1.134 VI_VI_CSIA_PATTERN_GEN_SIZE_0

Control register for CSI-A pattern generator

Offset: 0xa5 | Read/Write: R/W | Reset: 0x00000000 (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)

27.1.135 VI_VI_CSIA_PATTERN_GEN_CONTROL_0

Offset: 0xa6 | Read/Write: R/W | Reset: 0x00000002 (0b00xxxx0010)

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

27.1.136 VI_VI_CSIB_PATTERN_GEN_SIZE_0

Control Register for CSI-B pattern generator

Offset: 0xa7 | Read/Write: R/W | Reset: 0x00000000 (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)

27.1.137 VI_VI_CSIB_PATTERN_GEN_CONTROL_0

Offset: 0xa8 | Read/Write: R/W | Reset: 0x00000002 (0b00xxxx0010)

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

27.2 MIPI-CSI Registers

Refer to the CSI section in this document for descriptions of these registers.

[THIS PAGE INTENTIONALLY LEFT BLANK]

28.0 SD/MMC CONTROLLER

The SD/MMC Controller can interface with SD, SDIO, and eMMC devices. Tegra® 3 contains four instances of this controller:

Controller	UHS-I Signaling Supported	8-bit eMMC Supported	Signaling Voltages
SDMMC1	Yes	1 & 4-bit only	1.8, 2.8-3.3 V
SDMMC2	No	Yes	1.2, 1.8, 2.8-3.3 V
SDMMC3	Yes	Yes	1.8, 2.8-3.3 V
SDMMC4	No	Yes	1.2, 1.8, 2.8-3.3 V

28.1 Standards Supported

- "SD Specifications Part E1 SDIO Specification", Technical Committee SD Card Association, Version 2.00, 30-January 2007.
- "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 4.41, JEDEC SOLID STATE TECHNOLOGY ASSOCIATION, March 2010.

28.2 Speeds Supported

The following SD 3.0 data transfer modes are supported over the 4-bit interface:

- Default Speed mode: 3.3V signaling, Frequency up to 25 MHz, up to 12.5 MB/sec
- High Speed mode: 3.3V signaling, Frequency up to 50 MHz, up to 25 MB/sec
- SDR12: 1.8V signaling, Frequency up to 25 MHz, up to 12.5 MB/sec
- SDR25: 1.8V signaling, Frequency up to 50 MHz, up to 25 MB/sec
- SDR50: 1.8V signaling, Frequency up to 100 MHz, up to 50 MB/sec
- SDR104: 1.8V signaling, Frequency up to 208 MHz, up to 104 MB/sec
- DDR50: 1.8V signaling, Frequency up to 50 MHz, sampled on both clock edges, up to 50 MB/sec

28.3 Operation

28.3.1 Hardware / Software Partitioning

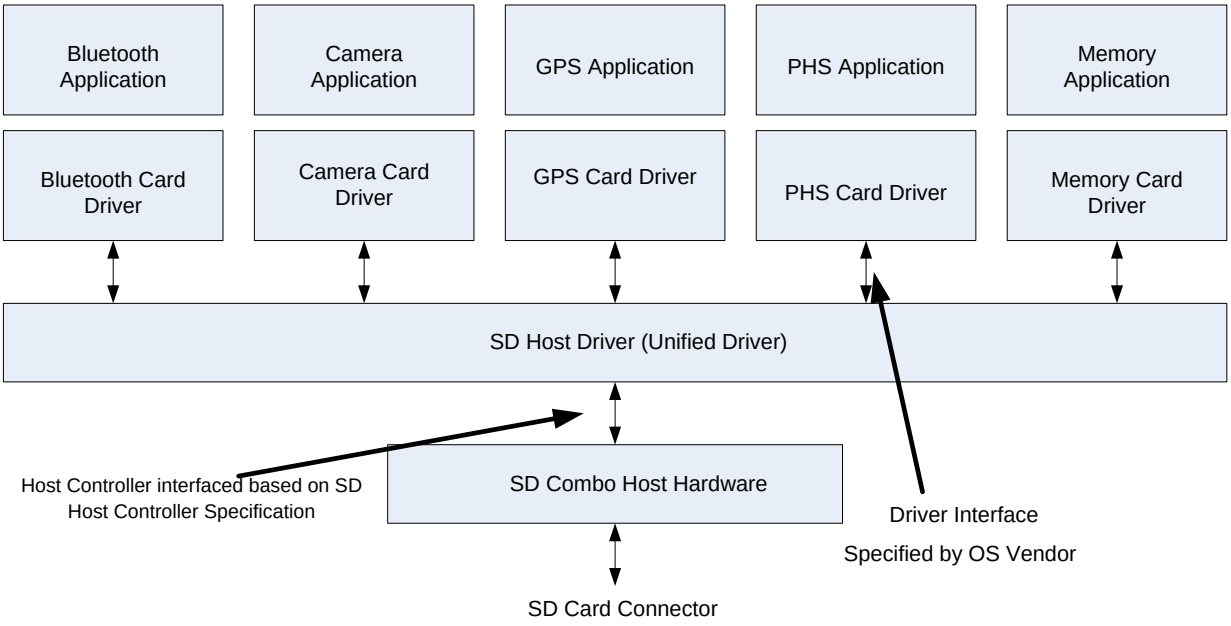
Hardware

Sends out the programmed command, stores the response, and makes 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

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 68. Host Hardware and Driver Architecture



28.4 Performance

Table 75 Performance Maximums

Interface Standard	Maximum Performance
SD & SDIO	Up to 104 MB/s in UHS-I SDR104 mode
eMMC	Up to 104 MB/s in 8-bit DDR mode

28.5 Caveats and Assumptions

- A single SDMMC controller handles only one device at a time.
- The software should configure SDMMC_VENDOR_CLOCK_CNTRL_x_SDMMC_CLK to _DISABLE before turning off the SDMMC clock (and configure back to _ENABLE after turning on the SDMMC clock). This is required in order to support asynchronous card interrupts when no clock is supplied to the card.
- In non-DMA mode of operation, the maximum block length supported is 512 bytes. Refer to the Programming Guidelines section for additional information.
- Tegra 3 PMC can be configured to wake Tegra 3 from the LP0 power state based on either:
  - Card Detection GPIO state
  - Asynchronous interrupt on the DAT1 line
- The SD Host Controller standard registers for reporting Card Detect and Write Protect state are tied off. Instead, software should check the Card Detect and Write Protect state through system-specific GPIOs.
- Off-card ECC, described in the MMC specification, is not supported.

28.6 Programming Guidelines

This section assumes a SD Host driver complying with the SD Specifications' Part A2, SD Host Control Standard Specification, is already available. The following subsections detail the necessary changes required for fully-featured SD (and eMMC boot mode) operation on Tegra 3 devices.

28.6.1 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:

28.6.1.1 General

These settings apply to all SDMMC controllers in use:

- Enable SD3.0 support by setting SDMMC_VENDOR_MISC_CNTRL_x_SDMMC_SPARE0[4].
- Enable SDR50 support by setting SDMMC_VENDOR_MISC_CNTRL_x_SDMMC_SPARE0[3].
- Enable tuning in SDR50 by setting SDMMC_VENDOR_CLOCK_CNTRL_x_SDR50_TUNING_OVERRIDE.
- Enable SDR104 support by setting SDMMC_VENDOR_MISC_CNTRL_x_SDMMC_SPARE0[2].
- Clamp clocks during resets by setting these two values:
  - SDMMC_VENDOR_CLOCK_CNTRL_x_PADPIPE_CLKEN_OVERRIDE_NORMAL
  - SDMMC_VENDOR_CLOCK_CNTRL_x_SPI_MODE_CLKEN_OVERRIDE_NORMAL

28.6.1.2 SDMMC1

These settings are specific to the SDMMC1 controller:

- If using the SDMMC1 interface, set the pads' drive strengths:
  - APB_MISC_GP_SDIO1CFGPADCTRL_0_CFG2TMC_SDIO1CFG_CAL_DRVDN = 6'h2A
  - APB_MISC_GP_SDIO1CFGPADCTRL_0_CFG2TMC_SDIO1CFG_CAL_DRVUP = 6'h2E
  - APB_MISC_GP_SDIO1CFGPADCTRL_0_CFG2TMC_SDIO1CFG_CAL_DRVDN_SLWR = 2'h1
  - APB_MISC_GP_SDIO1CFGPADCTRL_0_CFG2TMC_SDIO1CFG_CAL_DRVUP_SLWF = 2'h1
  - 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'h70
  - SDMMC_AUTO_CAL_CONFIG_0_AUTO_CAL_PU_OFFSET = 8'h62
- 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 pullup 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_PULL_UP_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT2_0_PULL_UP_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT1_0_PULL_UP_PULL_UP
  - PINMUX_AUX_SDMMC1_DAT0_0_PULL_UP_PULL_UP

28.6.1.3 SDMMC2

These settings are specific to the SDMMC2 controller:

- If using the SDMMC2 controller's external interface shared with the VI pins, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_VI_PCLK_0_PULL_UP_NORMAL
  - If using eMMC (a stronger external 4.7 kOHm pullup is used on the CMD line):
    - PINMUX_AUX_VI_D1_0_PULL_UP_NORMAL
  - If using SD or SDIO:
    - PINMUX_AUX_VI_D1_0_PULL_UP_PULL_UP
  - PINMUX_AUX_VI_D2_0_PULL_UP_PULL_UP
  - PINMUX_AUX_VI_D3_0_PULL_UP_PULL_UP
  - PINMUX_AUX_VI_D4_0_PULL_UP_PULL_UP
  - PINMUX_AUX_VI_D5_0_PULL_UP_PULL_UP
  - PINMUX_AUX_VI_D6_0_PULL_UP_PULL_UP
  - PINMUX_AUX_VI_D7_0_PULL_UP_PULL_UP
  - PINMUX_AUX_VI_D8_0_PULL_UP_PULL_UP
  - PINMUX_AUX_VI_D9_0_PULL_UP_PULL_UP
- If using the SDMMC2 controller's external interface shared with the KBC pins, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_KB_ROW10_0_PULL_UP_NORMAL
  - If using eMMC (a stronger external 4.7 kOHm pullup is used on the CMD line):
    - PINMUX_AUX_KB_ROW11_0_PULL_UP_NORMAL
  - If using SD or SDIO:
    - PINMUX_AUX_KB_ROW11_0_PULL_UP_PULL_UP
  - PINMUX_AUX_KB_ROW6_0_PULL_UP_PULL_UP
  - PINMUX_AUX_KB_ROW7_0_PULL_UP_PULL_UP
  - PINMUX_AUX_KB_ROW8_0_PULL_UP_PULL_UP
  - PINMUX_AUX_KB_ROW9_0_PULL_UP_PULL_UP
  - PINMUX_AUX_KB_ROW12_0_PULL_UP_PULL_UP
  - PINMUX_AUX_KB_ROW13_0_PULL_UP_PULL_UP
  - PINMUX_AUX_KB_ROW14_0_PULL_UP_PULL_UP
  - PINMUX_AUX_KB_ROW15_0_PULL_UP_PULL_UP
- If using the SDMMC2 controller's external interface shared with the DAP pins, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_DAP1_SCLK_0_PULL_UP_NORMAL
  - If using eMMC (a stronger external 4.7 kOHm pullup is used on the CMD line):
    - PINMUX_AUX_DAP1_FS_0_PULL_UP_NORMAL
  - If using SD or SDIO:
    - PINMUX_AUX_DAP1_FS_0_PULL_UP_PULL_UP
  - PINMUX_AUX_DAP1_DIN_0_PULL_UP_PULL_UP
  - PINMUX_AUX_DAP1_DOUT_0_PULL_UP_PULL_UP

- PINMUX_AUX_SPDIF_OUT_0_PULL_UP_PULL_UP
- PINMUX_AUX_SPDIF_IN_0_PULL_UP_PULL_UP

28.6.1.4 SDMMC3

These settings are specific to the SDMMC3 controller:

- If using the SDMMC3 interface, set the pads' drive strengths:
  - APB_MISC_GP_SDIO2CFGPADCTRL_0_CFG2TMC_SDIO2CFG_CAL_DRVDN = 6'h2A
  - APB_MISC_GP_SDIO2CFGPADCTRL_0_CFG2TMC_SDIO2CFG_CAL_DRVUP = 6'h2E
  - APB_MISC_GP_SDIO2CFGPADCTRL_0_CFG2TMC_SDIO2CFG_CAL_DRVDN_SLWR = 2'h1
  - APB_MISC_GP_SDIO2CFGPADCTRL_0_CFG2TMC_SDIO2CFG_CAL_DRVUP_SLWF = 2'h1
  - APB_MISC_GP_SDIO2CFGPADCTRL_0_CFG2TMC_SDIO3CFG_CAL_DRVDN = 6'h2A
  - APB_MISC_GP_SDIO2CFGPADCTRL_0_CFG2TMC_SDIO3CFG_CAL_DRVUP = 6'h2E
  - APB_MISC_GP_SDIO2CFGPADCTRL_0_CFG2TMC_SDIO3CFG_CAL_DRVDN_SLWR = 2'h1
  - APB_MISC_GP_SDIO2CFGPADCTRL_0_CFG2TMC_SDIO3CFG_CAL_DRVUP_SLWF = 2'h1
  - SDMMC_SDMEMCOMPPADCTRL_2_SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL = 4'h7
- If using SDMMC3 interface for SD or SDIO (instead of eMMC), additionally calibrate the pads by setting:
  - SDMMC_AUTO_CAL_CONFIG_2_AUTO_CAL_ENABLE_ENABLED
  - SDMMC_AUTO_CAL_CONFIG_2_AUTO_CAL_PD_OFFSET = 8'h70
  - SDMMC_AUTO_CAL_CONFIG_2_AUTO_CAL_PU_OFFSET = 8'h62
- If using the SDMMC3 interface, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_SDMMC3_CLK_0_PULL_UP_NORMAL
  - If using eMMC (a stronger external 4.7 kOHm pullup is used on the CMD line):
    - PINMUX_AUX_SDMMC3_CMD_0_PULL_UP_NORMAL
  - If using SD or SDIO:
    - PINMUX_AUX_SDMMC3_CMD_0_PULL_UP_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT3_0_PULL_UP_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT2_0_PULL_UP_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT1_0_PULL_UP_PULL_UP
  - PINMUX_AUX_SDMMC3_DAT0_0_PULL_UP_PULL_UP

28.6.1.5 SDMMC4

These settings are specific to the SDMMC4 controller:

- If using the SDMMC4 controller's external interface shared with the GMI pins, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_SDMMC4_CLK_0_PULL_UP_NORMAL
  - If using eMMC (a stronger external 4.7 kOHm pullup is used on the CMD line):
    - PINMUX_AUX_SDMMC4_CMD_0_PULL_UP_NORMAL
  - If using SD or SDIO:
    - PINMUX_AUX_SDMMC4_CMD_0_PULL_UP_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
- PINMUX_AUX_SDMMC4_RST_N_0_PULL_UP_PULL_DOWN

- If using the SDMMC4 controller's external interface shared with the CAM pins, correctly enable the integrated pullup/down resistors there:
  - PINMUX_AUX_CAM_MCLK_0_PULL_UP_NORMAL
  - If using eMMC (a stronger external 4.7 kOHm pullup is used on the CMD line):
    - PINMUX_AUX_GPIO_PCC1_0_PULL_UP_NORMAL
  - If using SD or SDIO:
    - PINMUX_AUX_GPIO_PCC1_0_PULL_UP_PULL_UP
  - PINMUX_AUX_GPIO_PBB0_0_PULL_UP_PULL_UP
  - PINMUX_AUX_CAM_I2C_SCL_0_PULL_UP_PULL_UP
  - PINMUX_AUX_CAM_I2C_SDA_0_PULL_UP_PULL_UP
  - PINMUX_AUX_GPIO_PBB3_0_PULL_UP_PULL_UP
  - PINMUX_AUX_GPIO_PBB4_0_PULL_UP_PULL_UP
  - PINMUX_AUX_GPIO_PBB5_0_PULL_UP_PULL_UP
  - PINMUX_AUX_GPIO_PBB6_0_PULL_UP_PULL_UP
  - PINMUX_AUX_GPIO_PBB7_0_PULL_UP_PULL_UP
  - PINMUX_AUX_GPIO_PCC2_0_PULL_UP_PULL_DOWN

28.6.2 eMMC Boot Modes

Note:	Only two of the four SD/MMC controllers can be a boot device: either SDMMC3 or SDMMC4.

28.6.3 Boot Option1

1. The clock to the card needs to be configured to 20 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 3 devices do not support ADMA modes during boot option 1.)
5. Configure **SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0** – The max 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 sec. The value programmed should exclude the 74 cycles 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_BUS[4].** Software shouldn't program commands until this bit is zero.

28.6.4 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.

28.6.5 Card Detect and Write Protect

The standard SD Host SDCD# and CDWP# registers act as if those two pins are connected to zero. So, the SD driver must get the SDCD# and CDWP# interrupts and status via platform-specific GPIOs.

Additionally, an abort command should be issued if a card is removed during an SD transfer.

28.6.6 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_x'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.

28.6.7 SD3.0 Signal Voltage Switching

Note: This section is applicable only to the UHS-I SDMMC1 and SDMMC3 controllers. The SDMMC2 and SDMMC4 controllers do not require these settings.

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 or SDMMC3 I/O voltage from 3.3 to 1.8V. This is typically via I2C control of an external "PMIC" (Power Management IC), but varies from platform to platform.

Similarly, 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 IO voltage back to 3.3V.

28.6.8 Tuning

Because Tegra 3 SDMMC 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. On Tegra 3 devices, 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 is issued to the card and wait for the Buffer Read Ready interrupt. Once the interrupt is generated, in HOST_CONTROL_2 register (offset 03Eh), check whether EXECUTE_TUNING bit is cleared and the SAMPLING_CLOCK_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.

28.7 SDMMC Registers

28.7.1 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:

28.7.1.1 Present State Register

Offset: 024h | Read/Write: R

Bit	Reset	Description
28:25	X	DAT_7_4_LINE_LEVEL: This status reflects the DAT[7:4] lines' state for debug.

28.7.1.2 Interrupt Status Register

Offset: 030h | Read/Write: R/W

Bit	Reset	Description
31:30	0	VEND_SPEC_ERROR 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.

28.7.1.3 Interrupt Status Enable Register

Offset: 034h | Read/Write: R/W

Bit	Reset	Description
31:30	0	VEND_SPEC_ERROR 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.

28.7.1.4 Interrupt Signal Enable Register

Offset: 038h | Read/Write: R/W

Bit	Reset	Description
31:30	0	VEND_SPEC_ERROR 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.

28.7.1.5 Force Event Register

Offset: 050h | Read/Write: R/W

Bit	Reset	Description
31:30	0	VEND_SPEC_ERROR 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.

28.7.2 Vendor-Specific Registers

The following Registers are Vendor Specific Registers and are mapped to the Vendor Specific Address Space (offsets 0x100 - 0x1FF).

28.7.2.1 SDMMC_VENDOR_CLOCK_CNTRL_0

TAP_VAL - Tap Value for the input data path trimmer. This determines the tap value needed to sample the input data correctly. Delay per each tap can range from 100ps to 250ps. Following values are recommended on different SDMMC interfaces for all modes up to SDR50.

- SDMMC1 - 5
- SDMMC2_DAP - 5
- SDMMC2_KBC - 5
- SDMMC2_VI - 6
- SDMMC3 - 5
- SDMMC4_PLANAR - 5
- SDMMC4_POP - 5

For SDR104 (only supported for SDMMC1 and SDMMC3), 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 SDMMC controller. This should be done after every time SDMMC is reset and after every soft reset.

This is important as all SDMMC 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).

Vendor Clock Control Register

Offset: 100h | Read/Write: R/W | Reset: 0b00000000001011101000 0xx001101

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

Bit	Reset	Description
		SDMMC1 and SDMMC3) 0 = NORMAL : 0 -> No Tuning support advertised for SDR50 mode. 1 = OVERRIDE : 1 -> Tuning support is enabled for SDR50 mode
4	0x0	HW_RSTN_OVERRIDE: Hardware reset override for SDMMC4 instance This is a sticky 1 bit, will be reset using PMC2CAR_RSTN reset. This bit is valid only for SDMMC4 Instance 0 = NORMAL : 0 -> No override. 1 = OVERRIDE : 1 -> Override enabled.
3	0x1	PADPIPE_CLKEN_OVERRIDE: Override for padmacro and pipemacro clken. 1 -> CLKEN is kept asserted to padmacro and pipemacro when internal CLKEN is de-asserted. 0 -> CLKEN is to padmacro and pipemacro can be de-asserted along with internal CLKEN
2	0x1	SPI_MODE_CLKEN_OVERRIDE: Override for CLKEN during SPI_MODE during sw_reset. 1 -> CLKEN is kept asserted while doing sw_reset. 0 -> CLKEN is de-asserted while doing sw_reset
1	0x0	INPUT_IO_CLK: Feedback clock is selected by default. Software should not change this. Disabling Feedback clock will select Internal Clock that requires 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'. 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. 0 = DISABLE 1 = ENABLE

28.7.2.2 SDMMC_VENDOR_SPI_CNTRL_0

Vendor SPI Control Register

Offset: 104h | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	0x0	SPI_MODE: SPI mode is not supported. 0 = DISABLE : 0 -> SPI mode is not enabled. 1 = Reserved.

28.7.2.3 SDMMC_VENDOR_SPI_INTR_STATUS_0

The fields are valid when a SPI error has occurred.

SPI Interrupt Status Register

Offset: 108h | Read/Write: RO | Reset: 0bxxxxxxxxx

Bit	Reset	Description
8:5	X	DAT_ERR_TOKEN: Data Error Token, while read from card
4:0	X	DAT_RESPONSE: Data Response while write to card 5 = DATA_ACCEPTED 11 = CRC_ERR 13 = WRITE_ERR

28.7.2.4 SDMMC_VENDOR_BOOT_CNTRL_0

This register is used to configure Boot Mode to support MMC v4.3 cards.

Vendor Boot Control Register

Offset: 110h | Read/Write: R/W | Reset: 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 enable, HW auto clears it when boot data is done. Writing 0 terminates the BootOption1 0 = DISABLE 1 = ENABLE

28.7.2.5 SDMMC_VENDOR_BOOT_ACK_TIMEOUT_0

Vendor Boot Acknowledgment Timeout Register

Offset: 114h | Read/Write: R/W | Reset: 0b00000000000000000000

Bit	Reset	Description
19:0	0x0	VALUE: If Boot Acknowledgment is not received within the programmed number of cycles. Boot Acknowledgement Timeout error occurs(VENDOR_SPECIFIC_ERR[0])

28.7.2.6 SDMMC_VENDOR_BOOT_DAT_TIMEOUT_0

Boot Data Timeout Register

Offset: 118h | Read/Write: R/W | Reset: 0b00000000000000000000000000

Bit	Reset	Description
24:0	0x0	VALUE: If Boot Data is not received within the programmed number of cycles. Then Data Timeout error occurs.

28.7.2.7 SDMMC_VENDOR_DEBOUNCE_COUNT_0

The Debounce Counter runs on 32 KHz clock.

Keeping the default value to 100ms = (100 * 32cycles/1ms) = 3200 cycles for 100ms = 0xC80

Debounce Counter Value Register

Offset: 11ch | Read/Write: R/W | Reset: 0b00000000000000110010000000

Bit	Reset	Description
23:0	0xc80	VALUE: The number of 32KHz clock cycles is programmed to meet Debounce period of the card slot.

28.7.2.8 SDMMC_VENDOR_MISC_CNTRL_0

- 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[4] : When set, SD3.0 support is advertised in SDMMC_SLOT_INTERRUPT_STATUS_0_SPECIFICATION_VERSION_NUMBER Otherwise, only SD2.0 support is advertised
  - 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_SPARE1: Spare register bits with reset value of 1
  - SDMMC_SPARE1[0] : CMD_CONFLICT_DISABLE, disable command line conflict if set to 1.

Misc Vendor Cntrl Register

Offset: 120h | Read/Write: R/W | Reset: 0b11111111111111110000000000000000

Bit	Reset	Description
31:16	0xffff	SDMMC_SPARE1: Spare register bits with reset value of 1
15:1	0x0	SDMMC_SPARE0: Spare register bits with reset value of 0
0	0x0	ERASE_TIMEOUT_LIMIT: Erase timeout value 0= Finite, It is limited to the programmed value in the DATA_TIMEOUT_VALUE 1= Infinite, Controller would be monitoring until the card is busy.

28.7.2.9 SDMMC_SDMEMCOMPPADCTRL_0

If AUTO_CAL_ENABLE is disabled (0), the values in this register is used to drive the DRVUP/DRVDN controls to the SDMMC pads.

SDMEMCOMP Pad control register

Offset: 1e0h | Read/Write: R/W | Reset: 0b1111111x1111111xxxxxxxx1000

Bit	Reset	Description
26:20	0x7f	CFG_SDMEMCOMP_DRVUP: used if AUTOCAL is disabled
18:12	0x7f	CFG_SDMEMCOMP_DRVDN: used if AUTOCAL is disabled
3:0	0x8	SDMMC2TMC_CFG_SDMEMCOMP_VREF_SEL

28.7.2.10 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: 1e4h | Read/Write: R/W | Reset: 0b0000xxxxxxxxx001x0000000x0000000

Bit	R/W	Reset	Description
31	RO	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	RW	0x0	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
29	RW	DISABLED	AUTO_CAL_ENABLE: 1 (normal operation): use SDMMC generated pullup/dn (override or autocal) 0 (disabled): use sdmmc2tmc_cfg* register settings for pullup/dn 0 = DISABLED 1 = ENABLED
28	RW	0x0	AUTO_CAL_SLW_OVERRIDE: AUTOCAL slew rate 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
18:16	RW	0x1	AUTO_CAL_STEP: calibration step interval (in microseconds)
14:8	RW	0x0	AUTO_CAL_PD_OFFSET: 2's complement offset for pull-down value
6:0	RW	0x0	AUTO_CAL_PU_OFFSET: 2's complement offset for pull-up value

28.7.2.11 SDMMC_AUTO_CAL_INTERVAL_0

SDMEMCOMP pad calibration interval - Valid for SDMMC1/SDMMC3 Instances

Offset: 1e8h | Read/Write: R/W | Reset: 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.

28.7.2.12 SDMMC_AUTO_CAL_STATUS_0

SDMEMCOMP pad calibration status - Valid for SDMMC1/SDMMC3 Instances

Offset: 1ech | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	AUTO_CAL_ACTIVE: One when auto calibrate is active - valid only after auto calibrate

Bit	Reset	Description
		sequence has completed (AUTO_CAL_ACTIVE == 0)
30:24	X	AUTO_CAL_PULLDOWN_ADJ: Pulldown code sent to pads
22:16	X	AUTO_CAL_PULLUP_ADJ: Pullup code sent to pads
14:8	X	AUTO_CAL_PULLDOWN: Pulldown code generated by auto-calibration
6:0	X	AUTO_CAL_PULLUP: Pullup code generated by auto-calibration

29.0 NAND FLASH CONTROLLER

The NAND flash interface controller allows Tegra® 3 devices to access NAND flash memories for mass storage. The controller supports both asynchronous (legacy) interfaces and ONFI 2.2 standard with error protection scheme required for data integrity in SLC and MLC devices. Tegra 3 devices support 24-bit BCH ECC correction. Up to 8 chip selects are supported natively, with more possible through GPIOs. The NAND Flash controller is an AHB master and can initiate high speed data transfers between external DRAM and NAND Flash memory through hardware buffering. NAND controller registers are connected on the APB interface as a slave, for register read/write access.

29.1 Features

The NAND Flash controller supports both PIO and DMA modes of operations for NAND Flash access. Interrupts are generated upon a COMMAND sequence completion, as programmed in the NAND_COMMAND register. Supported operations include individual cycles of command, address, read data and write data; or a combined sequence for program, erase and read operations. When hardware ECC is enabled, data transfer cycles are extended to update the flash spare area to store the parity data information.

In PIO transfer modes, data written into the RESP register is transmitted out, and similarly read data is captured in the RESP register. Typically, READ ID or READ STATUS sequences should make use of PIO mode of operation. PIO mode is not recommended for large transfers or when performance is a criterion.

Note:	PIO mode is not recommended for large transfers or when performance is a criterion.

In DMA mode, write data from DRAM is transmitted out, while for read transfers, data received from flash is transferred to DRAM as programmed in the DMA configuration registers. For each of the data pipelines, a separate FIFO is used to throttle the Read/Write transfers.

The NAND controller supports HW_ERROR_CORRECTION without using large buffers for page size data storage in the ECC decoder/encoder. It supports Error correction without using large buffers for page size data storage in ECC decoder/Encoder. Entire pages are not stored. Only sectors (512b/1kb chunks) are stored, thus reducing RAM area and increasing throughput.

The automatic RBSY status mode of operation reduces the software overhead by checking RBSY line status for CMB_RBSY application.

The NAND Controller supports Command queue mode of operation targeted to reduce the software overhead in managing the IO transfers on page basis. A number of Flash operations can be queued in this mode of operation with NAND DMA interface that will bring in these commands and execute them in sequence.

29.1.1 Feature Summary

- Supports asynchronous interface requirements of various NAND Flash memory vendors
- Programmable cycles to support multiple commands/operations in single sequence
- Programmable timing interface
- 8/16-bit data interface support
- Optional BCH coding based ECC for main/spare area data – 512 byte/1 kbyte codeword
- Programmable error correction capability of t=4,8,14, 16 and 24 bit error for each 512/1k bytes of data for BCH selection of main and spare area
- Programmable page size ranging from 512 bytes, 1k, 2k, 4k and 8k bytes
- EDO mode of read access for maximum read throughput
- Automated hardware status check

- Command queue mode of operation to queue up a number of flash operations
- Interface support up to 8 different devices
- Single DMA based programming interface for multi-page read/write transfer
- Scatter gather DMA structure that enables software to program fragmentation up front. The hardware automatically pre-fetches this information, and gathers fragments, thus eliminating the need for software to join the memory for non-contiguous memory targets
- Software control to interleave operations on multiple cards for optimum performance
- Extended decode status information to support software wear leveling and bad block management

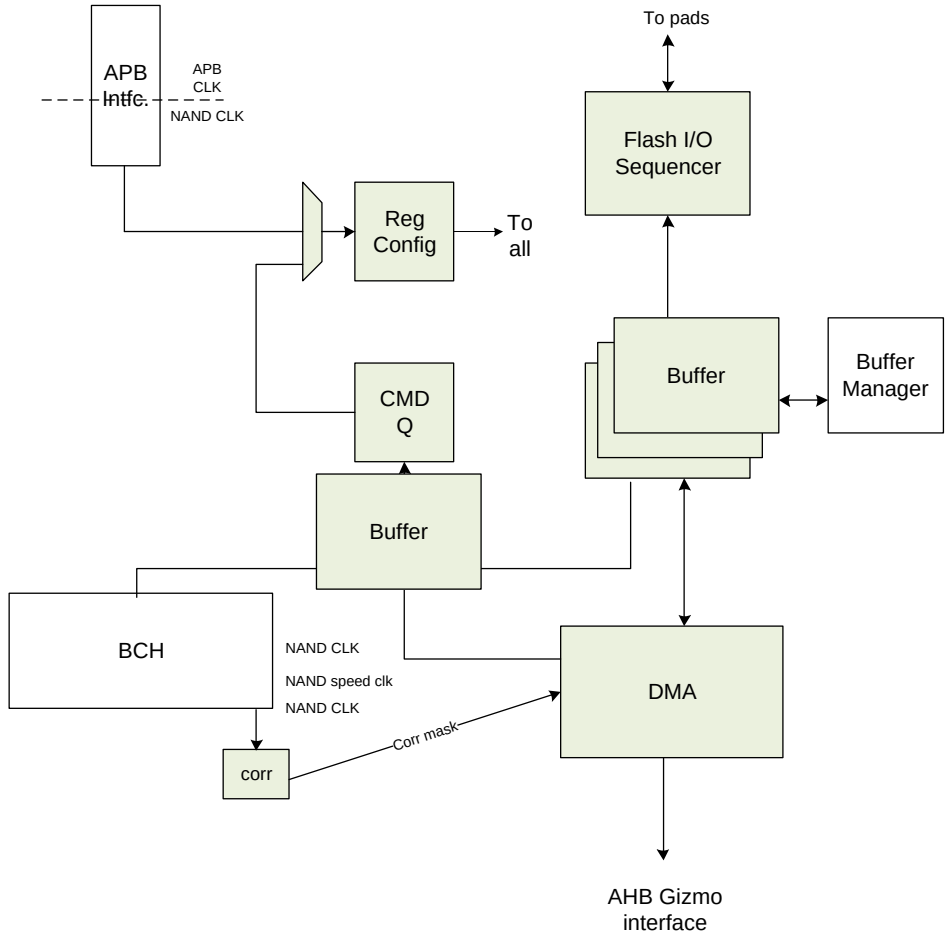
29.1.2 Software Features

The following Flash commands are supported:

- Flash Program (write)
- Flash Read
- Single plane Cache Write
- Double plane Cache Write
- Flash Cache Read
- Block Erase
- Read Status
- Read Status Multi-plane
- Reset
- Random Data Input/Output

29.2 Functional Description

Figure 69. NAND Controller Block Diagram



29.2.1 BCH Error Detection/Correction for Main and Spare Block Data

The following table summarizes the parity size lengths requirements for different Tag sizes.

Table 76 BCH ECC for Main and Spare

ECC (per 512 bytes)	Page Size (bytes)	Parity Size (bytes)
t=4	2k	28 bytes
t=8	2k	52 bytes
t=14	2k	92 bytes
t=16	2k	104 bytes
t=24	2k	104 bytes
t=4	4k	56 bytes
t=8	4k	104 bytes
t=14	4k	184 bytes

29.2.2 Hardware Error Correction

The NAND controller consists of quad buffers of 1k sector size to support hardware error correction. When ECC is enabled during data read from the NAND controller, data written to memory is already corrected (if it is correctable). Data read from the Flash interface is stored in the buffer and supplied to ECC logic to locate error location and apply correction on-the-fly with bit masks at the error bit locations while writing to memory.

29.2.3 Extended Decode Status Information

This section provides detailed information of error pattern development to enable software evaluation of wear-leaving and early detection of bad blocks. Once DMA completes a programmed number of pages of read, information is available of error pattern on a page basis. Without this mechanism, there is no way for software to figure out error pattern development in media, since hardware error correction is deployed. Once the errors approach the maximum correctable number, software can relocate these blocks to a new location.

Decode status buffer is stored with information of sub-page indication of correctable errors and un-correctable errors. For every page with correctable/uncorrectable error, an entry is made to this buffer. Upon DMA completion, software can read these buffer contents through APB register interface for error pattern analysis. Detailed description of related registers:

- CORRFAIL_ERR flag in NAND_ISR register indicates that DMA transfer of specified pages results in correctable OR un-correctable errors. If there are no correctable/un-correctable errors in any of the pages of DMA transfer, this bit is not SET.
- Register NAND_DEC_RESULT indicates the count of pages that result in correctable/un-correctable errors. Software reads these entries from decode status buffer.
- Register NAND_DEC_STAT_BUF. Reading this register fetches an entry from decode status buffer. Each entry of this buffer relates to page which results in either correctable/un-correctable error.

For both 8k page selection and 8k page selection with Tag only transfer enabled, there are 2 entries per page.

When page decode results in more number of errors than the threshold programmed OR uncorrectable errors, an entry is made into decode status buffer and an interrupt is raised. There are a couple of options available for software in this context depending on the use case:

- Service the interrupt read status buffer contents and Initiate abort.
- Service the interrupt read status buffer and continue the Read operation. Save the decode result statistics.

Do not enable/service the interrupt and let the DMA complete. Maximum number of page decode statistics are limited to 32 entries. This translates to 16 pages of DMA for 8k bytes page selection and 32 pages of DMA for all page sizes less than 8 kbytes.

Example

If 4 pages result in correctable errors and 2 pages in uncorrectable errors out of a total DMA transfer of 64 pages, PAGE_COUNT in NAND_DEC_RESULT will indicate as ‘PAGE_COUNT=6’. Software should read these 6 entries reading NAND_DEC_STAT_BUF register by CPU access and process it for further action.

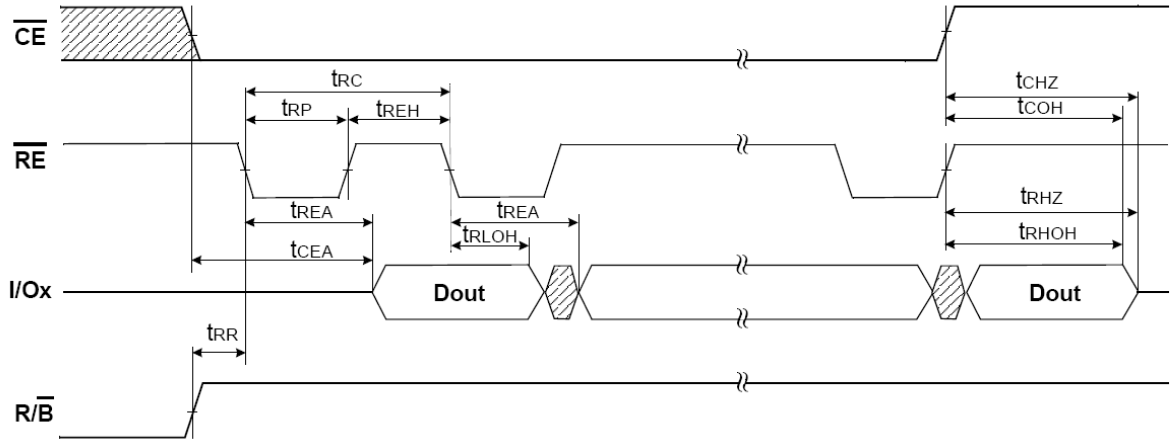
29.2.3.1 EDO Mode of READ Access

The latest generations of Flash support EDO (Extended Data Output) mode of operation. In this mode, Flash drives the data for an extended amount of time even after Read cycle hold time is completed. This enables the Host system to sample the data even on completion of READ pulse.

Since the typical read cycle of operation consists of asynchronous mode of operation (Host drives the READ pulse, and flash device drives the data), path/pad delays of read pulse extend the read pulse width so that correct data is sampled on the positive edge of READ pulse. Data availability at the following negative edge enables the Host system application to run without extending these pulse width timings, and thus improve performance.

EDO mode of read operation is controlled through the configuration bit EDO_MODE. Otherwise default Read data is sampled on pos-edge of READ pulse. The following diagram shows the extended data output timings.

Figure 70 EDO Mode Read Access

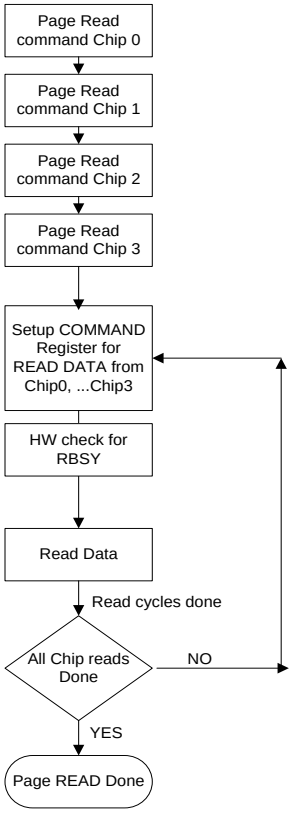


29.2.3.2 Automatic RBSY Status

The aim is to avoid software overhead checking the RBSY line status for CMB_RBSY application with single RBSY line. Read status command register NAND_HWSTATUS_CMD [7000:0050] is used for READ STATUS command issued by HW, configurable for the flash type used.

So when actual page read commands are issued, software can avoid the polling check for RBSY lines. When CMB_RBSY is enabled for read data transfer, hardware issues Read Status command using command byte in NAND_HWSTATUS_CMD register, and iterates till the flash is ready and moves to data read phase and completes the data transfer.

Figure 71 Interleaved READ Access for Combined RBSY Application with Auto RBSY



29.2.3.3 Skip Spare Bytes

In previous implementations of NAND Controllers with HW encoder, Tag data is written to flash in sequential order in spare area of flash. If main page ECC encode is not enabled, Tag data starts at location '0' in the spare area. Typically, with HW Encoder enabled, spare area write starts with parity bytes of main data. Flash vendors generally provide bad block information at these locations (there is no standard as to how much). These bytes need to be preserved for file system maintenance.

Skip spare feature allows achieving this functionality of skipping first few bytes of spare area as selected with SKIP_SPARE_SEL. Refer to NAND_CONFIG register for different options provided. Bit [23] of NAND_CONFIG, SKIP_SPARE_EN – turns on this new mode of operation. Ability to skip 4/8/12/16 bytes is supported with this enhancement.

29.2.3.4 Command Queue Processor

Command queue scheme is targeted to reduce the software overhead in managing the I/O transfers on a page basis. Each IO transfer typically consists of a page size of transfer for read/write/copy back or block erase. Considering 64k block of data transfer, current approach requires setup, initiation and track of 32 I/O CMD transfers for 2k page size selection. DMA is initialized once followed by multiple commands of I/O, sometimes repeatedly by just issuing the new command changing the address registers.

In command queue implementation, parameters required for each of the command (I/O transaction) are put together in the form of a packet. Software keeps on posting these packets, reducing the overhead to handle page transfer interrupts etc. NAND I/O and state control are now fed with the configuration of the current command in progress. After the completion of current command new packet is fetched from the command queue to proceed for the new I/O transfer without software intervention.

The following parameters are identified, which could vary from packet to packet depending on the flash sequences to be carried out. Each command packet consists few or all of the following parameters and each parameter is a 32-bit configuration register in the existing hardware implementation.

Table 77 Command Queue Registers

Command Queue Data set
NAND_COMMAND
NAND_CMD_REG1
NAND_CMD_REG2
NAND_ADDR_REG1
NAND_ADDR_REG2
NAND_DMA_MST_CTRL
NAND_DMA_CFG.A
NAND_DMA_CFG.B
NAND_DATA_BLOCK_PTR
NAND_TAG_PTR
NAND_ECC_PTR
NAND_HWSTATUS_CMD
NAND_HWSTATUS_MASK
NAND_CONFIG

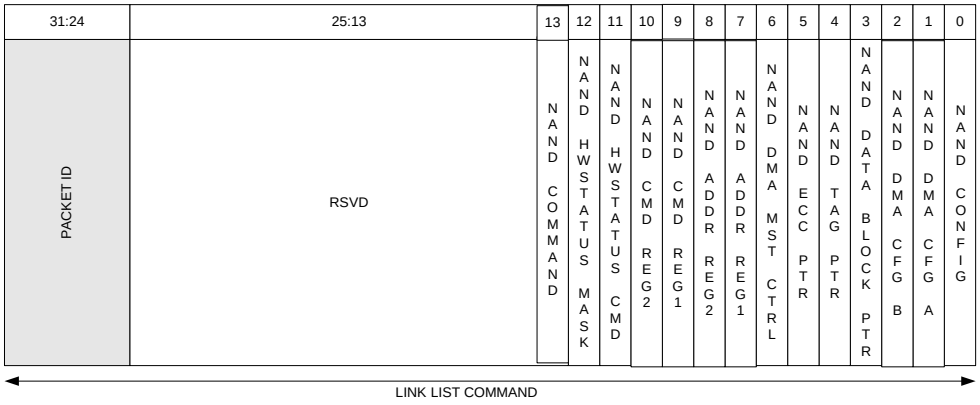
Note: Registers not listed above are not part of command queue configuration, and static during the duration of Command queue execution. These should be properly configured as described in programming guide lines and register spec.

Command Queue Definition

Terminology:

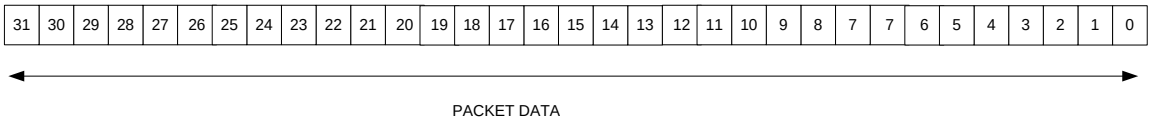
- Command Queue Command: command queue command register to define the register set programming.
- Command Queue Packet: one set of command and respective command data/registers.
- Command Queue Data: command packet parameter/register set value to be configured.
- Command Queue: set of command packets.

Figure 72 Command Queue Command Format



Within the ***command*** register, bit [13:0] is used to identify which register(s) need to be programmed with ***command queue data(s)*** that follows. A '1' within bit [13:0] indicates the corresponding register requires programming with ***command queue data***; a '0' indicates there's no programming needed for the corresponding register. The command queue command parser will always scan the ***command queue command*** starting from bit [0] to bit [13].

Figure 73 Command Queue Data Format



Following the ***command queue command***, one or more ***command queue data(s)*** follow. The 1st ***command queue data*** following the ***command queue command*** corresponds to the right most '1' bit within bit [13:0] of the ***command queue command***. Similarly, the 2nd command queue data following the command queue command corresponds to the 2nd right most '1' bit within bit [13:0] of the ***command queue command***.

Figure 74 Command Queue Packet Format

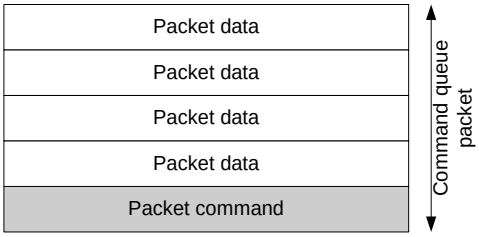
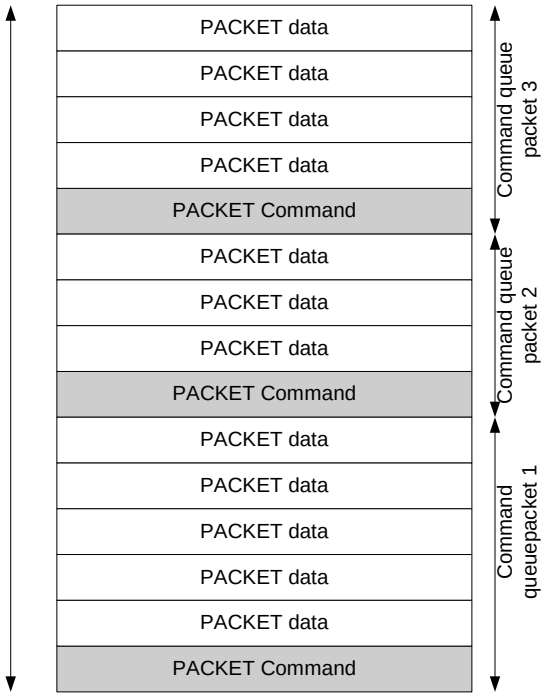


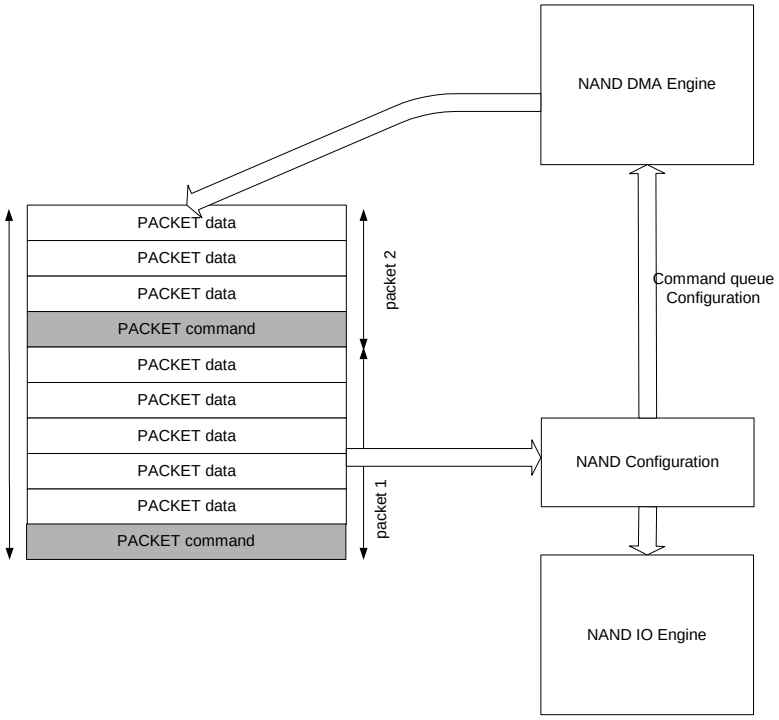
Figure 75 Command Queue Format



Command Queue Parser

Command queue parser fetches one command, followed by as many respective packet data fetches from this internal buffer as bit fields set in PACKET command register. When all the PACKET command data are fetched, corresponding sequence of register settings will configure the NAND Controller engine for specific sequence of IO operation. As soon as one command packet is completed, new command packet is processed.

Figure 76. Command Queue Parser



29.2.3.5 Memory Lock

The hardware provides a programmable address range on the flash controller, and a write once unchangeable bit for each range (unless the HW is reset) to implement this feature for software.

- NAND_LOCK_CONTROL: software configures which range check is applicable now.
- NAND_LOCK_STATUS: HW indicates which of the range is matched caused this interrupt.
- NAND_LOCK_APERTURE_START0 – NAND_LOCK_APERTURE_END0: If Block erase command address matches this range (inclusive of start, ends), actual flash erase is skipped and LOCK_ERROR interrupt is generated.

29.3 Programming Guidelines

29.3.1 Clocking

For Async or legacy single data rate mode:

- Configure CLK_SOURCE_NDFLASH register with desired clock source selection and DIVISOR settings like any peripheral clock programming requirements. NAND Controller clocks should be enabled with CLK_ENABLE_NDFLASH=1 it field in CLK_OUT_ENABLE_L register
- Configure CLK_SOURCE_NDFLASH register bit field NDFLASH_DIV2_SEL =0.

For DDR rate mode of interface:

- Configure CLK_SOURCE_NDFLASH (0x60006160) register with desired clock source selection and DIVISOR settings like any peripheral clock programming requirements. Frequency selection should be such that achieved clock frequency is twice the desired DDR interface frequency of operation.
- Configure CLK_SOURCE_NDFLASH register bit field NDFLASH_DIV2_SEL=1. This will make internal logic to operate on interface frequency and portion of logic on twice the interface frequency.

> **Note:** In Tegra 3 NAND, separate clocking branch has been added for clocking search algorithm in error correction logic. This is to speed up error location search during decode operation. Clock source options similar to CLK_SOURCE_NDFLASH exist for this clock branch.

- NAND speed clock should be enabled with CLK_ENABLE_NAND_SPEED=1 field in CLK_OUT_ENABLE_U register.
- Configure CLK_SOURCE_NAND_SPEED register with desired clock source selection and DIVISOR settings.
- NAND speed clock achieved frequency should be greater than or equal to NAND_CLK as programmed through CLK_SOURCE_NDFLASH register. For BootRom purpose, this may be programmed same as NAND_CLK where the decode latency is not a major concern.

29.3.2 Pin Muxing

Tegra 3 NAND has 2 pin multiplexing options, one on GMI pins and the other on KBC pins. The set of MISC registers to be programmed for both these selections are listed below.

29.3.2.1 GMI Option

1) With 8 Chip selects and 1 RBSY line.

PINMUX_AUX_GMI_IORDY_0 = 0x21
PINMUX_AUX_GMI_ADV_N_0 = 0x21
PINMUX_AUX_GMI_CS4_N_0 = 0x21
PINMUX_AUX_GMI_CS3_N_0 = 0x21
PINMUX_AUX_GMI_AD1_0 = 0x21
PINMUX_AUX_GMI_AD3_0 = 0x21
PINMUX_AUX_GMI_AD5_0 = 0x21
PINMUX_AUX_GMI_AD7_0 = 0x21
PINMUX_AUX_GMI_AD9_0 = 0x21
PINMUX_AUX_GMI_AD11_0 = 0x21
PINMUX_AUX_GMI_AD13_0 = 0x21
PINMUX_AUX_GMI_AD15_0 = 0x21
PINMUX_AUX_GMI_OE_N_0 = 0x21
PINMUX_AUX_GMI_WP_N_0 = 0x21
PINMUX_AUX_GMI_CS1_N_0 = 0x21
PINMUX_AUX_GMI_CS7_N_0 = 0x21
PINMUX_AUX_GMI_WAIT_0 = 0x21
PINMUX_AUX_GMI_CLK_0 = 0x21
PINMUX_AUX_GMI_CS2_N_0 = 0x21
PINMUX_AUX_GMI_AD0_0 = 0x21
PINMUX_AUX_GMI_AD2_0 = 0x21
PINMUX_AUX_GMI_AD4_0 = 0x21
PINMUX_AUX_GMI_AD6_0 = 0x21
PINMUX_AUX_GMI_AD8_0 = 0x21
PINMUX_AUX_GMI_AD10_0 = 0x21
PINMUX_AUX_GMI_AD12_0 = 0x21
PINMUX_AUX_GMI_AD14_0 = 0x21
PINMUX_AUX_GMI_WR_N_0 = 0x21
PINMUX_AUX_GMI_DQS_0 = 0x21
PINMUX_AUX_GMI_CS0_N_0 = 0x21
PINMUX_AUX_GMI_CS6_N_0 = 0x21
PINMUX_AUX_GMI_RST_N_0 = 0x21

2) With 4 Chip selects and 4 RBSY lines.

PINMUX_AUX_GMI_IORDY_0 = 0x21
PINMUX_AUX_GMI_ADV_N_0 = 0x21
PINMUX_AUX_GMI_CS4_N_0 = 0x21
PINMUX_AUX_GMI_CS3_N_0 = 0x21
PINMUX_AUX_GMI_AD1_0 = 0x21
PINMUX_AUX_GMI_AD3_0 = 0x21
PINMUX_AUX_GMI_AD5_0 = 0x21
PINMUX_AUX_GMI_AD7_0 = 0x21
PINMUX_AUX_GMI_AD9_0 = 0x21
PINMUX_AUX_GMI_WAIT_0 = 0x21
PINMUX_AUX_GMI_CLK_0 = 0x21
PINMUX_AUX_GMI_CS2_N_0 = 0x21
PINMUX_AUX_GMI_AD0_0 = 0x21
PINMUX_AUX_GMI_AD2_0 = 0x21
PINMUX_AUX_GMI_AD4_0 = 0x21
PINMUX_AUX_GMI_AD6_0 = 0x21
PINMUX_AUX_GMI_AD8_0 = 0x21
PINMUX_AUX_GMI_AD10_0 = 0x21

PINMUX_AUX_GMI_AD11_0 = 0x21
PINMUX_AUX_GMI_AD12_0 = 0x21
PINMUX_AUX_GMI_AD13_0 = 0x21
PINMUX_AUX_GMI_AD14_0 = 0x21
PINMUX_AUX_GMI_AD15_0 = 0x21
PINMUX_AUX_GMI_WR_N_0 = 0x21
PINMUX_AUX_GMI_OE_N_0 = 0x21
PINMUX_AUX_GMI_DQS_0 = 0x21
PINMUX_AUX_GMI_WP_N_0 = 0x21
PINMUX_AUX_GMI_CS0_N_0 = 0x21
PINMUX_AUX_GMI_CS1_N_0 = 0x21
PINMUX_AUX_GMI_CS6_N_0 = 0x20
PINMUX_AUX_GMI_CS7_N_0 = 0x20
PINMUX_AUX_GMI_RST_N_0 = 0x20

Also, NAND_FBIO_CFG [7000:811C] register should be programmed for either of these CS/RBSY selection.

- CFG_RETIME_STAGES=0x0

29.3.2.2 KBC Option

1) With 2 Chip selects and 1 RBSY line

PINMUX_AUX_KB_ROW0_0 = 0x21
PINMUX_AUX_KB_ROW1_0 = 0x21
PINMUX_AUX_KB_ROW2_0 = 0x21
PINMUX_AUX_KB_ROW3_0 = 0x21
PINMUX_AUX_KB_ROW4_0 = 0x21
PINMUX_AUX_KB_ROW5_0 = 0x21
PINMUX_AUX_KB_ROW6_0 = 0x21
PINMUX_AUX_KB_ROW7_0 = 0x21
PINMUX_AUX_KB_ROW8_0 = 0x21
PINMUX_AUX_KB_ROW9_0 = 0x21
PINMUX_AUX_KB_ROW10_0 = 0x21
PINMUX_AUX_KB_ROW11_0 = 0x21
PINMUX_AUX_KB_ROW12_0 = 0x21
PINMUX_AUX_KB_ROW13_0 = 0x21
PINMUX_AUX_KB_ROW14_0 = 0x21
PINMUX_AUX_KB_ROW15_0 = 0x21
PINMUX_AUX_KB_COL0_0 = 0x21
PINMUX_AUX_KB_COL1_0 = 0x21
PINMUX_AUX_KB_COL2_0 = 0x21
PINMUX_AUX_KB_COL3_0 = 0x21
PINMUX_AUX_KB_COL4_0 = 0x21
PINMUX_AUX_KB_COL5_0 = 0x21
PINMUX_AUX_KB_COL6_0 = 0x21
PINMUX_AUX_KB_COL7_0 = 0x21

NAND_FBIO_CFG [7000:811C] register should also be programmed

- CFG_RETIME_STAGES=0x1

29.3.3 Interrupts

For write transfers,

- DMA done interrupt, IS.DMA_DONE in NAND_DMA_MST_CTRL is set and it happens before the Command completion for a single page DMA transfer size.
- Command done interrupt, IS.CMD_DONE as in NAND_ISR indicates the I/O transaction completion as per NAND_COMMAND register configuration.
- If spare access with B_VALID or HW_ECC is enabled, CMD_DONE interrupt guarantees that all of these operations are completed.

For read transfers,

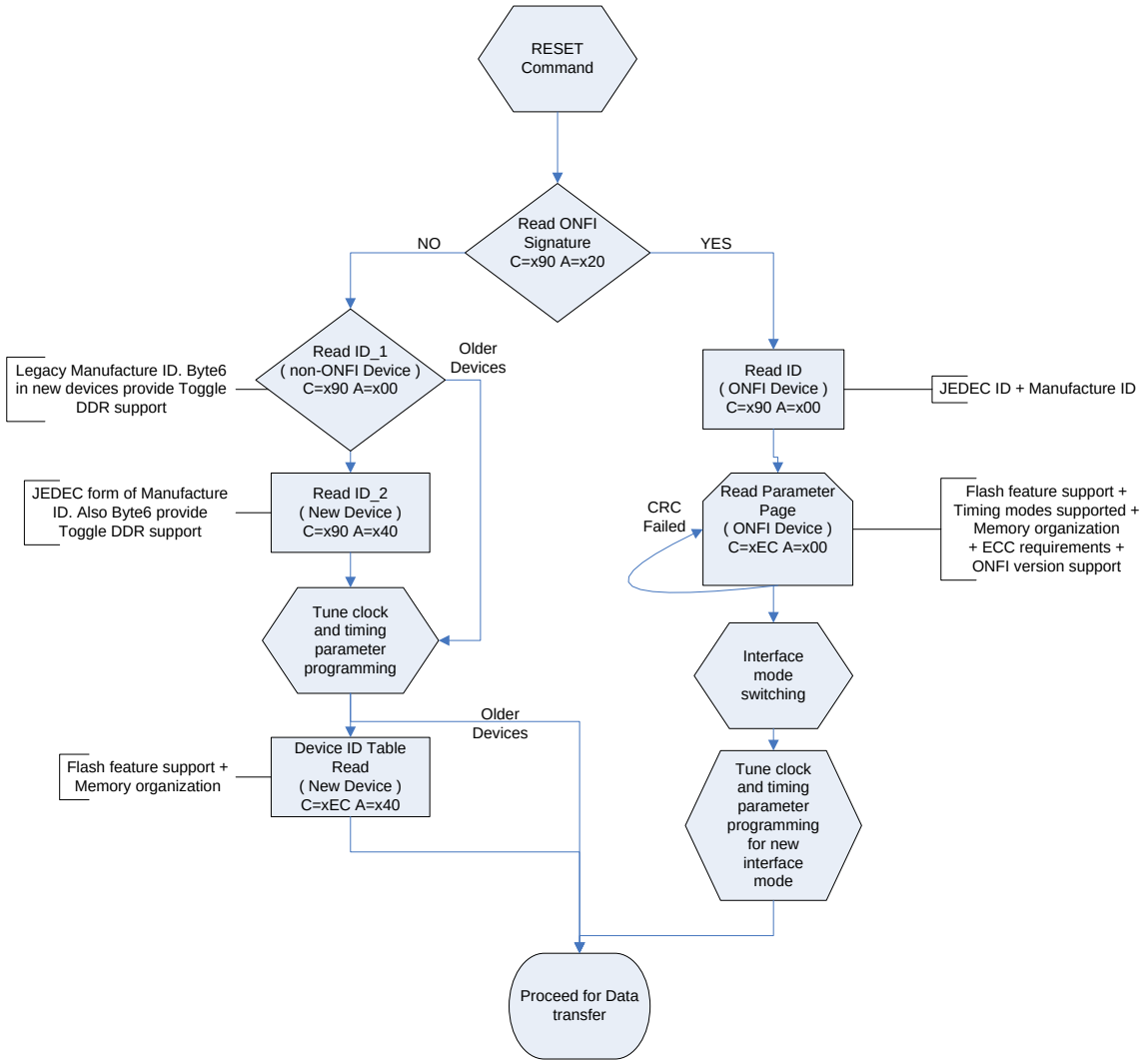
- Command done interrupt, IS.CMD_DONE as in NAND_ISR indicates the I/O transaction completion as per NAND_COMMAND register configuration.
- DMA done interrupt, IS.DMA_DONE in NAND_DMA_MST_CTRL is set which happens after the Command completion for a single page DMA transfer size. In addition it is guaranteed that actual data are transferred to XMB destinations as programmed which is inclusive of TAG information if it is enabled.

29.3.4 Device Initialization Sequence

The flow chart below details the sequence required during initialization to support variety of devices, i.e. starting from legacy NAND devices with asynchronous mode support, asynchronous devices with toggle DDR support, to all flavors of ONFI standards up to ONFI 2.2 specification.

- Once it is detected that the device is not ONFI compliant, from device maker code and device code, software has to figure out whether read ID information of 6 byte is valid or not. The latest generation devices from most vendors have 6th byte for interface mode capabilities. Bit fields in 6th byte indicate support of toggle DDR mode.
- For older legacy devices, device ID table read command may not be supported.
- For ONFI devices, Read Parameter Page structure provides information of interface mode support and other device capabilities.

Figure 77. Device Initialization Sequence



29.3.5 Interface Modes and Switching Modes

The latest generations of Flash devices support different interface modes of operation. Considering the power and performance requirements, software can operate in any of the interface modes supported by the device. Tegra 3 NAND Controller supports different interface modes as offered by the device manufacturers. The sections that follow outline the

interface modes and required guidelines for switching modes within that category. Programming of timing registers is covered in subsequent section. In general, timing register changes or NAND clock frequency changes should be done when the controller is idle and no operations are in progress to Flash.

29.3.5.1 Legacy Asynchronous Interface Devices

These are devices with legacy asynchronous interface, and are not ONFI compliant. Mode switching is not allowed for such devices. However, minimum timing requirements must be met as per NAND Controller clock frequency and timing register programming.

- Tegra 3 NAND Controller supports 2 sets of timing registers. With NAND_TIMING, NAND_TIMING2 registers timing programming is backward compatible to Tegra 2 etc.
- New set of timing registers are selected with TIMING_SEL=1 bit field in NAND_CONFIG2 register.
- Relevant timing registers: NAND_ASYNC_TIMING0, NAND_ASYNC_TIMING1, NAND_ASYNC_TIMING2, NAND_ASYNC_TIMING3

29.3.5.2 Toggle DDR Interface Devices

In these devices, there is no interface mode switching as in ONFI devices.

- On power on, device comes up in toggle interface.
- Different interface mode speeds are achieved with NAND clock frequency selection. Timing parameters have to be worked out as per timing equations along with clock frequency selection.
- Enable toggle DDR interface mode of operation with TOGGLE_DDR=1 setting in NAND_CONFIG2 register
- Relevant timing registers: NAND_ASYNC_TIMING_0, NAND_ASYNC_TIMING_1, NAND_ASYNC_TIMING_2, NAND_ASYNC_TIMING_3, NAND_TDDR_TIMING_0, NAND_TDDR_TIMING_1

29.3.5.3 ONFI Devices (Including DDR Capable Devices)

Devices conforming to ONFI standards fall in this category. Devices compliant to ONFI standards can range from asynchronous mode of transfer to DDR mode of transfer. Interface mode switching is allowed in such devices as per guidelines indicated in the standard.

Summary

- On power on, device comes up in asynchronous mode 0
- Program timing registers to work for ONFI async mode 0 timing requirements
- Perform Device initialization routine as described in section 3.4.2 of ONFI 2.2 spec
- Issue Read Parameter Page command to identify device capabilities and interface modes supported
- Issue Set Features command to switch to async modes of higher speed or source synchronous (DDR) modes
- Configure clocking changes for NAND Controller to operate at new frequency as required. In case of switching from async mode to DDR mode, additional clock source selection changes are required as described in clocking section.
- Update NAND Controller timing registers as per new interface mode. For switching to async modes, update of respective register fields is required. Switching to DDR mode requires additional timing register configuration.
- Enable toggle DDR interface mode of operation with SYNC_DDR=1 setting in NAND_CONFIG2 register. For asynchronous interface mode only, SYNC_DDR=0 configuration is required.
- Relevant timing registers for asynchronous modes: NAND_ASYNC_TIMING_0, NAND_ASYNC_TIMING_1, NAND_ASYNC_TIMING_2, and NAND_ASYNC_TIMING_3.
- Relevant timing registers for source synchronous modes: NAND_ASYNC_TIMING_0, NAND_ASYNC_TIMING_1, NAND_ASYNC_TIMING_2, NAND_SDDR_TIMING_0, NAND_SDDR_TIMING_1

29.3.6 Programming of Command, Address and Data Transfer Sequences

NAND Flash Controller register interface allows for a flexible programming sequence that is fully under firmware control. Firmware can issue command, address and data transfers individually or combine them to form Read/Write/Status read sequences that the NAND flashes require.

For all the programming sequences described below, it is assumed that,

- NAND_TIMING register has already been programmed with the correct value for the NAND device in use. More details on programming the NAND timing register are given below.
- BUS_WIDTH field is set or cleared based on the flash card data width requirement.

29.3.6.1 Command Transfer

- Write to the NAND_CMD_REG1 with the command byte.
- Write to NAND_COMMAND register with CLE bit set, CE bit field set to the desired Chip Enable and the GO bit set. The CLE bit being set indicates that a command needs to be issued. The GO bit being set initiates the CLE transfer on the interface.
- Firmware can now check the status of the operation by polling the GO status. GO bit is cleared by the hardware on completion of the operation. Alternately, software can enable operation completion interrupt and handle the interrupt

29.3.6.2 Address Transfer

- Write to the NAND_ADDR_REG1/NAND_ADDR_REG2 with address bytes to be transferred. Software needs to form the address bytes in the format and sequence expected by the NAND device being used. Order of the address transfer by hardware is: NAND_ADDR_REG1 [7:0], NAND_ADDR_REG1 [15:8], NAND_ADDR_REG1 [23:16], NAND_ADDR_REG1 [31:24], NAND_ADDR_REG2 [7:0], NAND_ADDR_REG2 [15:8], NAND_ADDR_REG2 [23:16], NAND_ADDR_REG2 [31:24]. Actual number of address bytes transferred depends on the ALE_SIZE field. A maximum of 8 address bytes can be issued.
- Write to NAND_COMMAND register with ALE bit set, CE bit field set to the desired Chip Enable and the GO bit set. The ALE bit being set indicates that address cycles are present in the current sequence. The GO bit being set initiates the ALE transfer on the interface.
- Wait for completion of the operation through GO bit polling or CMD_DONE interrupt.

29.3.6.3 Write Data Transfer

This is used for writing data to the NAND Flash.

- Configure the PAGE_SIZE_SEL.A, BUS_WDITH fields in the NAND_CONFIG register as required by the Page size requirement of the Flash.
- Configure the DMA_BLOCK_SIZE.A field in the NAND_DMA_CFG.A register with the number of bytes to be transferred. This is the total number of bytes to be transferred and can be up to 64KBytes.
- Configure the NAND_DATA_BLOCK_PTR register with the data block address to be written to the flash.
- The above configuration initiates data transfers to flash. However, transfers to the device start as soon as the data is available in FIFO. Data transfers are throttled based on the internal buffer status.
- Program the NAND_DMA_MST_CTRL register DIR, BURST SIZE fields for transmit and enable the DMA_GO bit.
- Write to NAND_COMMAND register with the TX bit set, the CE bit field set of the desired chip and the GO bit set.
- This operation starts the write transfers on the NAND interface with command values from registers and data from buffers. Individual data transfer of byte or half-word wide is based on BUS_WIDTH parameter programmed. Transfers continue until a programmed size of data is written.

29.3.6.4 Read Data Transfers

This is used for reading data from the NAND Flash.

- Configure the DMA_BLOCK_SIZE.A and PAGE_SIZE_SEL.A fields in the NAND_DMA_CFG.A register with the number of bytes to be transferred. This is the total number of bytes to be transferred and can be up to 64KB.
- Configure the NAND_DATA_BLOCK_PTR register with the data block address to be written to the flash.
- The above configuration's initiates data reads from the device to internal buffer. Data reads on the NAND interface are throttled based on the buffer status.
- Program the NAND_DMA_MST_CTRL register DIR, BURST SIZE fields for receive and enable the DMA_GO bit. The above configuration prepares the transfer from buffers to AHB. But the actual data transfer starts only after valid entries available in buffers
- Write to NAND_COMMAND register with the RX bit set, the CE bit field set of the desired chip and the GO bit set.
- This Read operation continues until the transfer size worth of data is read and placed into the buffer as per transaction size selected. DMA Read operation continues until the buffer read is transferred to DMA destination of the programmed DMA length.
- Polling or interrupt can be used by firmware to determine the completion of each operation.

29.3.7 Programming of Flash Operations

All sequences described below are optimized for the performance of common NAND sequences by enabling multiple control bits when issuing NAND_COMMAND. Software has the flexibility of issuing these transfer operations individually and achieving the same results, albeit with lower performance.

Note:	In the following description, "operation" is used to indicate the command issued to the NAND Flash Controller; "sequence" refers to the sequence of events on the NAND Flash Interface.

29.3.7.1 Reset Sequence

In order to reset the NAND Flash, a reset command needs to be issued.

- Write to NAND_CMD_REG1 with LSB byte as "0xFF"
- Write to NAND_COMMAND register with CLE bit set, and GO bit set. Clear all the other control bits.
- Reset command is issued on the NAND interface
- NAND Controller operation completion is reflected in the status bit (GO bit is cleared). Additionally, command done interrupt is generated if it has been enabled.

29.3.7.2 Status Read Sequence

This sequence is used to read the status from the NAND Flash. By definition, this requires issuing a command "0x70" followed by reading a byte of data. This can be achieved in a single operation as follows:

- Write to NAND_CMD_REG1 with LSB byte as "0x70"
- Write to NAND_COMMAND register with the following
- CLE bit set
- PIO bit set, RX bit set
- TRANS_SIZE set to 1 byte
- Clear bits TX, SEC_CMD, AFT_DAT, A_VALID, B_VALID
- Set the GO bit
- The GO bit being set initiates the Controller "operation" to issue a sequence on the NAND interface.

- NAND controller sends out the CLE with command “0x70” and follows this with a response read of one byte from the NAND device. The Controller handles all the timing requirements of the NAND device based on the parameters programmed in NAND_TIMING register. The read byte is placed into the NAND_RESP register in little-endian format
- When the read is complete, NAND Controller operation is complete. This is reflected in the status bits, and command done interrupt is also generated when enabled. If polling is used, firmware can poll the GO bit. This is set by firmware to initiate the transfer and when the operation is done, hardware will clear this bit.

29.3.7.3 Read ID Sequence

Read ID requires issuing a READ ID command (0x90), followed by an address cycle and then reading 4 bytes of ID information. This is achieved as follows:

- Write to NAND_CMD_REG1 with LSB byte as “0x90”
- Write to NAND_ADDR_REG1 with the address byte.
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required.
  - PIO, RX bits are set
  - TX bits is cleared
  - TRANS_SIZE field is set to 4 bytes (maximum of 4 bytes is only possible)
  - SEC_CMD, AFT_DAT bits are cleared
  - Clear A_VALID, B_VALID bit
  - Set CE bit field for desired chip select.
  - Set GO bit starts interface command sequence.
- When the GO bit is set, CLE (0x90) -> ALE -> DATA READ sequence is initiated on the interface:
- Write to NAND_CONFIG register with the following
  - Set EDO_MODE field if flash memory supports. Note that tRP/tRP_RESP timing values should be adjusted to reduce the round trip delay factor, hence improving the performance.
  - Set BUS_WIDTH field for 16-bit data interface selection
- The data read from flash is written into the NAND_RESP register (since PIO bit is set)
- When entire sequence is completed, the operation done is indicated by the command done interrupt as well as hardware clear of ‘GO’ bit.

29.3.7.4 Block Erase Sequence

Block erase requires block erase setup command to be issued (0x60), followed by Address for the block (row only), and finally the block erase command. This is achieved as follows:

- Write to NAND_CMD_REG1 with LSB byte as “0x60”
- Write to NAND_CMD_REG2 with LSB byte as “0xD0”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the block address.
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required.
  - PIO, TX, RX bits are cleared
  - Set the SEC_CMD bit (for issuing the second command “0xD0”)
  - AFT_DAT bit is cleared (0), since the second command needs to be after address stage

- Clear A_VALID, B_VALID bits
- Set CE bit field for desired chip select.
- Set GO bit starts interface command sequence.

- When the GO bit is set , CLE (0x60) -> ALE –> CLE (0xD0) sequence is initiated on the interface
- When entire sequence is completed, the operation done is indicated by the command done interrupt as well as hardware clear of `GO' bit

29.3.7.5 Page Program Sequence

This sequence is used to write a page of data to the NAND Flash without ECC and without spare region access. First a write command “0x80” should be issued, followed by the destination address. Then, all the page data is written to the device. After writing the page data, programming is initiated by issuing the program command “0x10”. This sequence is achieved as follows:

- Write to NAND_CMD_REG1 with LSB byte as “0x80”
- Write to NAND_CMD_REG2 with LSB as “0x10”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct address bytes to be transferred.
- Configure NAND_TIMING, NAND_TIMING2 registers as described later in this document
- Write to NAND_CONFIG register with the following
  - Configure the PAGE_SIZE_SEL to select the page transfer size based on flash memory organization
  - If PROG_OFFSET_EN feature is disable, set SKIP_SPARE field so that controller can skip writing to factory bad block information at the start of spare region.
  - Set BUS_WIDTH field for 16-bit data interface selection
  - Set COM_RBSY field for wired-AND connection of flash RBSY lines to RBSY0 line.
- For COM_RBSY selection, configure NAND_HWSTATUS_CMD and NAND_HWSTATUS_MASK register appropriately. NAND controller uses this to complete the flash sequence by issuing read status commands to figure out the previous flash operation status.
- Configure NAND_DATA_BLOCK_PTR register with system buffer location to be programmed in main area
- Configure NAND_TAG_PTR register with system buffer location to be programmed in flash spare
- Configure the DMA_BLOCK_SIZE.A register with proper DMA transfer size required for a page transfer. However, for multi-page transfer this could be as much as the total DMA transfer size required.
- Configure NAND_DMA_CFG.B register with the total number of TAG bytes to be transferred from the spare region for all the 4 pages along with the TAG_BYTE_SIZE for a page. If the tag information is 8 bytes, then program this to (4 * 12 bytes/page) = 48 byte transfer.
- Configure NAND_IER and NAND_DMA_MST_CTRL registers as per the software requirements of interrupt indication on flash interface sequence or DMA sequence completion
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - TRANS_SIZE set to page size.
  - TX bit is set and PIO bit is cleared (indicates that this is a data should be fetched from FIFO)
  - Set the SEC_CMD bit (for issuing the second command “0x10”)
  - Set AFT_DAT bit since second command needs to be issued after the data stage.
  - Set A_VALID bit since only the data is being written. Note that spare region can also be written to by setting A_VALID and providing the appropriate address and data size information.
  - Set CE bit field for desired chip select

- Enable the DMA controller for transferring data to the NAND flash FIFO under flow control with the following in NAND_DMA_MST_CTRL register:
  - Set DMA_EN_A = 1
  - Set DMA_DIR=1 for Flash write operation
  - Select proper BURST_SIZE field per use case requirement.
  - Set DMA_GO field which starts DMA operation
- Start IO transfer by setting GO bit in NAND_COMMAND register, When the GO operation is initiated the NAND Controller issues the following sequence on the interface: CLE (0x80) -> ALE -> DATA -> CLE (0x10). The DATA stage is throttled based on the availability of data in the FIFO.
- When entire sequence is completed, the operation done is indicated by the command done interrupt as well as hardware clear of `GO' bit. DMA completion is indicated by hardware clear of `DMA_GO' bit.

29.3.7.6 Page Read Sequence

This sequence is used to read a page of data from the NAND Flash without ECC and without spare region access. First a read command “0x00” should be issued, followed by the destination address. Then, read is issued through command “0x30”. When the data becomes ready in the NAND Flash, the entire page is then read out. This sequence is achieved as follows:

- Write to NAND_CMD_REG1 with LSB byte as “0x00”
- Write to NAND_CMD_REG2 with LSB byte as “0x30”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct address bytes to be read.
- Configure the DMA_BLOCK_SIZE.A register with proper DMA read transfer size required for a PAGE transfer. However, for multi-page transfer this could be as much as the total DMA transfer size required.
- Configure the data block source address of read data in NAND_DATA_BLOCK_PTR register.
- Configure NAND_IER register as per the software requirements of interrupt indication on NAND IO sequence or DMA sequence completion
- Configure NAND_TIMING, NAND_TIMING2 registers as described later section in this document
- Write to NAND_CONFIG register with the following
  - Configure the PAGE_SIZE_SEL to select the page transfer size based on flash memory organization
  - If PROG_OFFSET_EN is disabled, set SKIP_SPARE field so that controller can skip writing to factory bad block information at the start of spare region.
  - Set EDO_MODE field if flash memory supports. Note that tRP/tRP_RESP timing values should be adjusted to reduce the round trip delay factor, hence improving the performance.
  - Set BUS_WIDTH field for 16-bit data interface selection
  - Set COM_RBSY field for wired-AND connection of flash RBSY lines to RBSY0 line.
  - Reading with ECC is covered in later sections with multiple pages
- For COM_RBSY selection, configure NAND_HWSTATUS_CMD and NAND_HWSTATUS_MASK register appropriately. NAND controller uses this to complete the flash sequence by issuing read status commands to figure out the previous operation status.
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - TRANS_SIZE set to page size selection with MSB set
  - RX bit is set and PIO bit cleared (indicates that this is a read data should be placed FIFO)
  - Set the SEC_CMD bit (for issuing the second command “0x30”)

  - AFT_DAT bit needs to be cleared since second command needs to be issued before the data stage.
  - Set A_VALID bit since only data is being read. Spare region can also be read by issuing multiple A_VALID commands with the appropriate address and size. .
- Set CE bit field for desired chip select
- Enable the DMA controller for transferring data to the NAND flash FIFO under flow control with the following in NAND_DMA_MST_CTRL register:
  - Set DMA_EN_A = 1
  - Set DMA_DIR=0 for Flash read operation
  - Select proper BURST_SIZE field per use case requirement.
  - Set DMA_GO field which starts DMA operation
- Start I/O transfer by setting GO bit in NAND_COMMAND register. When the GO operation is initiated the NAND Controller issues the following sequence on the interface: CLE (0x00) -> ALE -> CLE (0x30) -> DATA read. The DATA stage is throttled based on the availability of empty slots in the FIFO. Before entering the data stage the controller will ensure the NAND Flash is ready.
- When entire sequence is completed, the operation done is indicated by the command done interrupt as well as hardware clear of `GO' bit

29.3.7.7 Tag Only Write Sequence

This programming sequence describes the case of spare only region writes with and without ECC. The illustration is for a 4 page transfer. First a write command "0x80" should be issued, followed by the destination address. Then, all the spare data is written to the device. After writing the spare data, programming is initiated by issuing the program command "0x10". This sequence is achieved as follows:

- Write to NAND_CMD_REG1 with LSB byte as "0x80"
- Write to NAND_CMD_REG2 with LSB byte as "0x10"
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct row & column address byte location where the transfer starts.
- Write to NAND_CONFIG register with the following
  - Set BUS_WIDTH field for 16-bit data interface selection
  - Set COM_RBSY field for wired-AND connection of flash RBSY lines to RBSY0 line.
  - SKIP_SPARE feature bits do not have any significance.
  - Configure TAG_BYTE_SIZE inclusive of ECC parity bytes (fixed 4 bytes of parity for Tag information) if ECC option is selected. Note that (n-1) values should be programmed in TAG_BYTE_SIZE field for `n' bytes transfer
- Program NAND_DMA_CFG.B register with the total number of TAG bytes to be transferred to the spare region for all the 4 pages along with the TAG_BYTE_SIZE for a page. If the tag information is 8 bytes, then program this to (4 * 8bytes/page) = 32bytes transfer. Enabling ECC for write operation doesn't alter DMA length programming, since these are calculated and appended by hardware automatically. Note that (n-1) values should be programmed in DMA size field for `n' words/bytes transfer respectively.
- Configure NAND_TAG_PTR register with system buffer location to be programmed in flash
- Configure NAND_IER register as per the software requirements of interrupt handling.
- Configure NAND_TIMING, NAND_TIMING2 registers as described later section in this document
- For COM_RBSY selection or RBSY_CHK/RD_STATUS_CHK sequences, configure NAND_HWSTATUS_CMD and NAND_HWSTATUS_MASK registers appropriately. NAND controller uses these registers to complete the flash sequence by issuing read status commands automatically to figure out the previous operation status or flash is ready for the next operation, etc.

- Write to NAND_COMMAND register with the following:
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - TRANS_SIZE set to page size selection with MSB bit set.
  - TX bit is set (indicates that this is a data write)
  - Set the SEC_CMD bit (for issuing the second command “0x10”)
  - Set AFT_DAT bit since second command needs to be issued after the data stage
  - Set B_VALID=1 and A_VALID=0 to indicate that spare access is only required
- Set CE bit field for desired chip select
- For ECC case configure NAND_BCH_CONFIG register:
  - Set BCH_TAG_TVALUE field for error correction strength selection
  - Set BCH_ECC field to enable error correction
- Enable the DMA controller for transferring data to the device under flow control with the following in NAND_DMA_MST_CTRL register:
  - Set DMA_EN_B = 1
  - Set DMA_DIR=1 for Flash write operation
  - Select proper BURST_SIZE field per use case requirement.
  - Set DMA_GO field which starts DMA operation
- Start I/O transfer by setting GO bit in NAND_COMMAND register. When the GO operation is initiated the NAND Controller issues the following sequence on the interface: CLE (0x80) -> ALE -> WR Tag bytes -> WR Tag ECC bytes (if ECC is enabled) -> CLE (0x10). When entire sequence is completed, operation done is indicated by the command done interrupt as well as Hardware clear of `GO’ bit.
- Since the DMA transfer size was programmed for 4 pages, the DMA is constantly filling the FIFO with the next page data (and similarly the tag information). Therefore, the next page data is readily available in the FIFO at the end of the prior sequence.
- Repeat next page write operation by programming the GO bit in NAND_COMMAND register as described above. For this entire transfer of 4 pages spare data transfer, one DMA setup is required, followed by 4 commands being issued by writing to the NAND_COMMAND register for every page.
- DMA completion is indicated by hardware clear of `DMA_GO’ bit. When IO completes actual transfer to device GO bit in NAND_COMMAND register is cleared.

In Tegra 3 devices, hardware also allows positioning of parity bytes and tag information bytes in spare area. The additional programming requirement for this mode of operation is as follows:

- Mode is determined by setting PROG_OFFSET_EN =1 field in NAND_CONFIG2 register.
- NAND_ADDR1, NAND_ADDR2 registers be programmed to indicate row/page location in flash.
- Configure NAND_TAG_OFFSET and NAND_TAG_ECC_OFFSET registers to specify the respective locations in spare area. Hardware computes final column address based on above configuration plus PAGE_SIZE selection.
- Configure BCH_TAG_TVAL selection in NAND_BCH_CONFIG2 register since tag can have different error correction capability independent of main data selection.
- Configure NAND_RANDOM_WR register with Random Data Input command code appropriate to that device. Default value of 0x85 should work for wide variety of devices.

29.3.7.8 Tag Only Read Sequence

This programming sequence describes the case of spare only region reads with and without ECC. The illustration is for a 4 page transfer. First a read command “0x00” should be issued, followed by the destination address. Then, read is issued through command “0x30”. Tag only access with ECC is defined as programmable offset feature turned off with spare only access. This sequence is achieved as follows:

- Write to NAND_CMD_REG1 with LSB byte as “0x00”
- Write to NAND_CMD_REG2 with LSB byte as “0x30”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct row & column address byte location where the transfer starts.
- Write to NAND_CONFIG register with the following
  - Set BUS_WIDTH field for 16-bit data interface selection
  - Set COM_RBSY field for wired-AND connection of flash RBSY lines to RBSY0 line.
  - Set EDO_MODE field if flash memory supports. Note that tRP/tRP_RESP timing values should be adjusted to reduce the round trip delay factor, hence improving the performance.
  - SKIP_SPARE feature bits do not have any significance.
  - Configure TAG_BYTE_SIZE inclusive of ECC parity bytes (fixed 4 bytes of parity for Tag information) if ECC option is selected. Note that (n-1) values should be programmed in TAG_BYTE_SIZE field for `n’ bytes transfer
- For ECC case configure NAND_BCH_CONFIG register,
  - Set BCH_TAG_TVALUE field for error correction strength selection
  - Set BCH_ECC field to enable error correction
- Program NAND_DMA_CFG.B register with the total number of TAG bytes to be transferred from the spare region for all the 4 pages along with the TAG_BYTE_SIZE for a page. If the tag information is 8 bytes, then program this to (4 * 12 bytes/page) = 48 byte transfer. When ECC is enabled, Tag size for DMA configuration should consider the parity data size bytes, since these read by hardware automatically. Note that (n-1) values should be programmed in DMA size field for `n’ words/bytes transfer respectively.
- Configure NAND_TAG_PTR register with system buffer location to be programmed in flash
- Configure NAND_IER register as per the software requirements of interrupt handling.
- Configure NAND_TIMING, NAND_TIMING2 registers as described later section in this document
- For COM_RBSY selection or RBSY_CHK/RD_STATUS_CHK sequences, configure NAND_HWSTATUS_CMD and NAND_HWSTATUS_MASK registers appropriately. NAND controller uses these registers to complete the flash sequence by issuing read status commands automatically to figure out the previous operation status or flash is ready for next operation, etc.
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - TRANS_SIZE set to page size selection with MSB bit set.
  - RX bit is set (indicates that this is a data write)
  - Set the SEC_CMD bit (for issuing the second command “0x10”)
  - Set AFT_DAT=0 since second command needs to be issued to initiate flash data read
  - Set B_VALID =1 and A_VALID=0 to indicate that spare access is only required.
  - Set CE bit field for desired chip select
- Enable the DMA controller for transferring data from NAND controller internal buffer to DMA destination with the following in NAND_DMA_MST_CTRL register:

- Set DMA_DIR=0 for Flash read operation
- Set DMA_EN_B = 1
- Select proper BURST_SIZE field per use case requirement.
- Set DMA_GO field which starts DMA operation

- Start I/O transfer by setting GO bit in NAND_COMMAND register. When the GO operation is initiated the NAND Controller issues the following sequence on the interface: CLE (0x00) -> ALE -> CLE (0x30) > RD Tag bytes -> RD Tag Parity bytes.(with ecc enabled). When entire sequence is completed, it is indicated by the command done interrupt as well as hardware clear of `GO' bit.
- Since the DMA transfer size was programmed for 4 pages, the DMA is constantly waiting to pullout from buffers
- Software repeats next page read operation by programming the GO bit in NAND_COMMAND register as described above. For this entire transfer of 4 pages spare data transfer, one DMA setup is required, followed by 4 commands being issued by writing to the NAND_COMMAND register for every page.
- Since the DMA transfer waits for IO transfer, DMA_GO completes the total transfer

In Tegra 3 devices, hardware allows reading of parity bytes and tag information bytes in spare area. The additional programming requirements for this mode are as follows:

- Mode is determined by setting PROG_OFFSET_EN =1 field in NAND_CONFIG2 register.
- NAND_ADDR1, NAND_ADDR2 registers be programmed to indicate row/page location in flash.
- Configure NAND_TAG_OFFSET and NAND_TAG_ECC_OFFSET registers to specify the respective locations in spare area. Hardware computes final column address based on above configuration plus PAGE_SIZE selection.
- Configure BCH_TAG_TVAL selection in NAND_BCH_CONFIG2 register since tag can have different error correction capability independent of main data selection.
- Configure NAND_RANDOM_RD register with Random Data Output command codes appropriate to that device. Default value of 0x05, 0xEO should work for wide variety of devices.

29.3.7.9 Multiple Page Write with ECC for Data and Tag

This sequence describes the case where multiple pages of data are to be transferred to the NAND device, along with Tag information into the spare region with error protection. Block A refers to the data payload (usually a page size), and Block B refers to the TAG information of spare for description below. Tag bytes of spare region are treated as extended sector of main page data. Parity byte computation of last sector of page includes the tag bytes. Following sequence shows the required programming for transferring 4 pages of data, along with the associated tag.

- Write to NAND_CMD_REG1 with LSB byte as “0x80”
- Write to NAND_CMD_REG2 with LSB byte as “0x10”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct address bytes to be transferred.
- Configure NAND_TIMING, NAND_TIMING2 registers as described later section in this document.
- Write to NAND_CONFIG register with the following
  - Configure the PAGE_SIZE_SEL to select the page transfer size based on flash memory organization
  - Set SKIP_SPARE field so that controller can skip starting few bytes in spare region along with SKIP_SPARE_SEL setting. If SKIP_SPARE=0, controller starts writing the parity bytes at the column “0” of spare region.
  - Set BUS_WIDTH field for 16-bit data interface selection
  - Set COM_RBSY field for wired-AND connection of flash RBSY lines to RBSY0 line.
  - Configure TAG_BYTE_SIZE (n-1) value for `n' bytes transfer
  - Set EDO_MODE field if flash memory supports. Note that tRP/tRP_RESP timing values should be adjusted to reduce the round trip delay factor, hence improving the performance.

- For COM_RBSY selection, configure NAND_HWSTATUS_CMD and NAND_HWSTATUS_MASK register appropriately. NAND controller uses this to complete the flash sequence by issuing read status commands to figure out the previous operation status.
- Configure the Data block source address in NAND_DATA_BLOCK_PTR register.
- Configure the Tag data source address in NAND_TAG_PTR register.
- Program NAND_DMA_CFG.A register with proper DMA_BLOCK_SIZE.A (4*page_size) as per PAGE_SIZE_SEL fields.
- Program NAND_DMA_CFG.B register with the total number of TAG bytes to be transferred to the spare region for all the 4 pages along with the TAG_BYTE_SIZE for a page. If the tag information is 8 bytes, then program this to (4 * 8bytes/page) = 32bytes transfer. Please note that DMA size programming register actually works on (n-1) words programming for n word transfer.
- Enable ECC mechanism by setting BCH_ECC in NAND_BCH_CONFIG register. BCH_TVALUE field selects the ECC correction capability for Data and tag.
- Clear PROG_OFFSET_EN bit field in NAND_CONFIG2 register. This is the default selection on reset.
- Configure NAND_IER and NAND_DMA_MST_CTRL registers as per the software requirements of interrupt indication on flash interface sequence or DMA sequence completion
- Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - TRANS_SIZE set to page size selection with MSB bit set.
  - TX bit is set (indicates that this is a data write)
  - Set the SEC_CMD bit (for issuing the second command “0x10”)
  - Set AFT_DAT bit since second command needs to be issued after the data stage.
  - Set A_VALID, B_VALID. A_VALID being set indicates that this operation requires data payload transfer, B_VALID being set indicates that spare access required for Tag information.
  - Set CE bit field for desired chip select
- Enable the DMA controller for transferring data to the internal buffer under flow control with the following in NAND_DMA_MST_CTRL register:
  - Set DMA_DIR=1 for Flash write operation
  - Set DMA_EN_A = 1
  - Set DMA_EN_B = 1
  - Select proper BURST_SIZE field per use case requirement.
  - Set DMA_GO field which starts DMA operation
- Start I/O transfer by setting GO bit in NAND_COMMAND register.
- When entire sequence is completed, the operation done is indicated by the command done interrupt as well as Hardware clear of `GO’ bit.
- Since the DMA transfer size was programmed for 4 pages, the DMA is constantly filling the internal buffer with the next page data (and similarly the tag information). Therefore, the next page data is readily available as the current sequence is above to finish on flash interface.
- Repeat next page write operation by programming the NAND_COMMAND register. This process is repeated for the remaining pages. For this entire transfer of 4 pages, one DMA setup is required, followed by 4 commands being issued by writing to the NAND_COMMAND register for every page.

In Tegra 3 devices, hardware allows positioning of parity bytes and tag information bytes in spare area as per offset registers. The additional programming requirements for this mode of operation are as follows:

- Mode is determined by setting PROG_OFFSET_EN =1 field in NAND_CONFIG2 register.
- NAND_ADDR1, NAND_ADDR2 registers determine the starting address of the page
- Configure NAND_MAIN_ECC_OFFSET, NAND_TAG_OFFSET and NAND_TAG_ECC_OFFSET registers to specify the respective locations in spare area.
- Configure BCH_TAG_TVAL selection in NAND_BCH_CONFIG2 register since tag can have different error correction capability independent of main data selection.
- Configure NAND_RANDOM_WR register with Random Data Input command code appropriate to that device. Default value of 0x85 should work for wide variety of devices.

29.3.7.10 Multiple Page Read with ECC for Data and Tag

This sequence describes the case where multiple pages of data along with Tag information are to be read from the NAND device with error protection. Block A refers to the data payload (usually a page size), and Block B refers to the Tag information of Spare region contents for description. Tag bytes of spare region are treated as extended sector of main page data. Syndrome data computation of last sector of page includes the tag bytes. Following sequence shows the required programming for transferring 4 pages of data, along with the associated tag.

- Write to NAND_CMD_REG1 with LSB byte as “0x00”
- Write to NAND_CMD_REG2 with LSB byte as “0x30”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct address bytes to be read.
- Configure NAND_IER and NAND_DMA_MST_CTRL registers as per the software requirements of interrupt indication on flash interface sequence or DMA sequence completion
- Configure NAND_TIMING, NAND_TIMING2 registers as described later section in this document
- Write to NAND_CONFIG register with the following
  - Configure the PAGE_SIZE_SEL to select the page transfer size based on flash memory organization
  - Set SKIP_SPARE field so that controller can skip starting few bytes in spare region along with SKIP_SPARE_SEL setting. SKIP_SPARE programming is only required if spare access/ECC scheme is enabled
  - Set EDO_MODE field if flash memory supports. Note that tRP/tRP_RESP timing programming can be adjusted to reduce the round trip delay factor, hence improving the performance.
  - Set BUS_WIDTH field for 16-bit data interface selection
  - Set COM_RBSY field for wired-AND connection of flash RBSY lines to RBSY0 line.
  - Configure TAG_BYTE_SIZE (n-1) value for `n’ bytes transfer
- For COM_RBSY selection, configure NAND_HWSTATUS_CMD and NAND_HWSTATUS_MASK register appropriately. NAND controller uses this to complete the flash sequence by issuing read status commands to figure out the previous operation status.
- Clear PROG_OFFSET_EN bit field in NAND_CONFIG2 register. This is the default selection on reset.
- Enable ECC mechanism by setting BCH_ECC in NAND_BCH_CONFIG register. BCH_TVALUE field selects the ECC correction capability for Data and tag.
- Configure the Data block source address in NAND_DATA_BLOCK_PTR register.
- Configure the Tag data source address in NAND_TAG_PTR register.
- Program NAND_DMA_CFG.A register with proper DMA_BLOCK_SIZE.A (4*page_size) and PAGE_SIZE_SEL fields. Configure the Data block destination address in NAND_DATA_BLOCK_PTR register.
- Program NAND_DMA_CFG.B register with proper DMA_BLOCK_SIZE.B (4*TAG_BYTE_SIZE). Configure the Data block destination address in NAND_DATA_BLOCK_PTR register.
- Write to NAND_COMMAND register with the following
  - CLE bit set

- ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
- TRANS_SIZE set to page size.
- RX bit is set (indicates that this is a read)
- Set the SEC_CMD bit (for issuing the second command “0x30”)
- AFT_DAT bit needs to be cleared since second command needs to be issued before the data stage.
- Set A_VALID and B_VALID. A_VALID being set indicates that this operation requires data payload transfer and B_VALID indicates the Tag information to be transferred.
- CE bit field set to the desired chip.

- Enable the DMA controller for transferring data to the NAND flash FIFO under flow control with the following in NAND_DMA_MST_CTRL register:
  - Set DMA_DIR=0 for Flash read operation
  - Set DMA_EN_A = 1
  - Set DMA_EN_B = 1
  - Select proper BURST_SIZE field per use case requirement.
  - Set DMA_GO field which starts DMA operation
- Start IO transfer by setting GO bit in NAND_COMMAND register. When the GO operation is initiated the NAND Controller issues the following sequence on the interface: CLE (0x00) -> ALE -> CLE (0x30) ->RD DATA -> RD ECC Parity Bytes -> RD Tag Info. The DATA/Tag stage is throttled based on the availability of space in the respective FIFO’s.
- When entire sequence is completed, the operation done is indicated by the command done interrupt as well as Hardware clear of `GO’ bit.
- ECC correctable error or uncorrectable error as discovered in decode process are pushed in to decode status buffer on page basis. Once the entire DMA sequence is completed software can read this decode status buffer for error statistics
- Initiate next page read operation by setting the GO bit in the NAND_COMMAND register. This process is repeated for the remaining pages. For this entire transfer of 4 pages, one DMA setup is required, followed by 4 commands being issued by writing to the NAND_COMMAND register after every page.

In Tegra 3, hardware allows reading of parity bytes and tag information bytes in spare area as specified in offset registers. The additional programming requirements for this mode of operation are as follows:

- Mode is determined by setting PROG_OFFSET_EN =1 field in NAND_CONFIG2 register.
- NAND_ADDR1, NAND_ADDR2 registers determine the starting address of the page
- Configure NAND_MAIN_ECC_OFFSET, NAND_TAG_OFFSET and NAND_TAG_ECC_OFFSET registers to specify the respective locations in spare area.
- Configure BCH_TAG_TVAL selection in NAND_BCH_CONFIG2 register since tag can have different error correction capability independent of main data selection.
- Configure NAND_RANDOM_RD register with Random Data Output command codes appropriate to that device. Default value of 0x05, 0xEO should work for wide variety of devices.

29.3.7.11 PIO Write data transfers

This describes the case of write transfers for page program in PIO mode of operation. The following sequence on the interface is required in the interface to achieve this program sequence: CLE (0x80) -> ALE -> DATA -> CLE (0x10). This sequence can be viewed as multiple GO sequences in PIO mode such as,

- **Step 1:** Single GO command sequence for combined CLE, ALE transfers for initialization
- **Step 2:** Multiple GO commands to transfer data in PIO mode

- **Step 3:** Final program command

The programming sequence illustration assumes a page size of 512bytes without spare area access. For spare area access data sequence as in Step 2 should be repeated for required transfer length.

- **Step1:** Write to NAND_CMD_REG1 with LSB byte as “0x80”
- **Step1:** Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct address bytes to be transferred.
- **Step1:** Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - CE bit field set to the desired chip.
  - GO bit is set, which initiates the IO transfer and gets cleared on transfer completion.
- Write to NAND_RESP register with appropriate data bytes to be transferred. LSB byte in this register is the first one to be transmitted out. NAND_RESP registers is a shared register used for PIO write data/PIO read data. Note that NAND_RESP write data is not readable by software
- **Step2:** Write to NAND_COMMAND register with the following
  - Clear any other bit fields which are SET earlier
  - TRANS_SIZE set to select the small transaction sizes and configure the TRANS_SIZE to be 0x3 for 4 byte transfer. In PIO mode only 4 bytes can be transferred with the single GO command
  - TX bit is set and PIO bit is set which indicates that this is a data should be fetched from NAND_RESP register
  - Set A_VALID bit since only the data is being written. Note that spare region can also be written to by setting A_VALID and provide the appropriate address and data size information
  - CE bit field set to the desired chip
  - GO bit is set, which initiates the IO transfer and gets cleared on transfer completion
- **Step3:** Repeat the data transfer sequence multiple times(128 times for total 512 byte transfer) setting the GO bit to complete the total page transfer.
- **Step3:** Write to NAND_CMD_REG1 with LSB as “0x10”
- **Step3:** Write to NAND_COMMAND register with the following
  - Clear any other bit fields which are SET earlier
  - CLE bit set
  - CE bit field set to the desired chip.
  - GO bit is set, which initiates the I/O transfer and gets cleared on transfer completion.

29.3.7.12 PIO Read data transfers

This describes the case of read transfers for page read in PIO mode of operation. Following sequence on the interface is required in the interface to achieve this sequence: CLE (0x00) -> ALE -> CLE (0x30) -> DATA. This sequence can be viewed as multiple GO sequences in PIO mode such as,

- **Step1:** Single GO command sequence for combined CLE (0x00), ALE and CLE (0x30) transfers for initialization
- **Step2:** Wait for RBSY from flash
- **Step 2:** Multiple GO commands to transfer data in PIO mode.

Following programming sequence illustration assumes a page size of 512bytes without spare area access. For spare area access data sequence as in Step 2 should be repeated for required transfer length.

- Write to NAND_CMD_REG1 with LSB byte as “0x00” and NAND_CMD_REG2 with LSB byte as “0x30”
- Write to NAND_ADDR_REG1/NAND_ADDR_REG2 with the correct address bytes to be transferred.

- **Step1:** Write to NAND_COMMAND register with the following
  - CLE bit set
  - ALE bit set, ALE_SIZE field is set to the appropriate number of bytes required
  - Set the SEC_CMD bit (for issuing the second command “0x30”)
  - AFT_DAT bit needs to be set cleared since second command needs to be issued before data cycles
  - CE bit field set to the desired chip.
  - GO bit is set, which initiates the IO transfer and gets cleared on transfer completion.
- **Step2:** Wait for RBSY=1 status from flash. RDY status from flash indicates that data is copied from internal cells to page register. Host controller can pulse read cycles to read data now
- **Step3:** Write to NAND_CONFIG register with the following
  - Set BUS_WIDTH field for 16-bit data interface selection
  - Set COM_RBSY field for wired-AND connection of flash RBSY lines to RBSY0 line.
  - Set EDO_MODE field if flash memory supports. Note that tRP/tRP_RESP timing values should be adjusted to reduce the round trip delay factor, hence improving the performance.
- For COM_RBSY selection, configure NAND_HWSTATUS_CMD and NAND_HWSTATUS_MASK register appropriately. NAND controller uses this to complete the flash sequence by issuing read status commands to figure out the previous operation status.
- **Step3:** Write to NAND_COMMAND register with the following
  - Clear any other bit fields which are SET earlier
  - TRANS_SIZE set to select the small transaction sizes and configure the TRANS_SIZE to be 0x3 for 4 byte transfer. In PIO mode only 4 bytes can be transferred with the single GO command
  - RX bit is set and PIO bit is set which indicates that this is a data should be fetched from NAND_RESP register.
  - CE bit field set to the desired chip.
  - GO bit is set, which initiates the IO transfer and gets cleared on transfer completion
- Read data is available in NAND_RESP register after the GO command completion.
- Repeat the data transfer sequence multiple times(128 times for total 512 byte transfer) setting the GO bit to complete the total page transfer.

29.3.8 RBSY_CHK and RD_STATUS_CHK programming

RBSY_CHK and RD_STATUS_CHK have been introduced in NAND Controller to facilitate command queue mode, but these features can also be handy in normal mode and reduces good amount of interrupt handling overhead.

If RBSY_CHK is enabled in NAND_COMMAND register for a flash sequence:

NAND controller waits for the RBSY status of selected flash to start command/Address/data cycles. If COM_RBSY=0, RBSY lines are readily available to figure the flash ready status. If COM_RBSY=1, controller issues Read Status Command using the command code set in HWSTATUS_CMD register to figure out the same. On the received data, it applies the RBSY_MASK and compares the data with expected value. RBSY_EXP_VAL should be programmed as per the flash specification, including the polarity. If response data masked with RBSY_MASK_VAL matches, then it proceeds for actual Command/Address/Data cycle transfer. Otherwise it keeps on repeating the same procedure after pre-defined time. If RD_STATUS_CHK is also enabled, controller proceeds to RD_STATUS_CHK phase.

If RD_STATUS_CHK is enabled in NAND_COMMAND register for flash sequence:

NAND controller issues Read Status Command using the command code set in HWSTATUS_CMD register. On the received response data, it applies RD_STATUS_MASK and compares the data with expected value. If it doesn’t match with RDSTATUS_EXP_VAL it raises error interrupt ISR.LL_ERROR. RDSTATUS_EXP_VAL should be

programmed as per the flash specification. If response data masked with RDSTATUS_MASK_VAL matches, then it proceeds for actual Command/Address/Data cycle transfer.

29.3.9 ECC Performance/ power knobs

In Tegra 3 devices, a system of monitoring the pipeline bottleneck is supported. A part of the BCH pipeline which is usually the bottleneck, is the polynomial root search engine. This logic can be potentially clocked at a higher rate than the rest of the BCH engine and NAND module. The rest of the logic operates at 100/200 MHz (nand_clk) while the search engine can operate at any frequency

Nand_clk <= nand_speed_clk <- 400 MHz.

Software has the option of dynamically setting the nand_speed_clk.

There are a few hardware register to help Software make this decision. Both these registers are relevant only for Read with ECC decode enabled operation.

The Buf_stall_accumulated registers aggregates the number of cycles that the I/O engine has stalled due to all the buffers being full. The corr_fifo_accumulated register counts the stalls due to the search engine. Software can look at the Buf_stall register to find out whether the NAND module is back pressuring the I/O interface during an ECC read, and compare the corr_fifo register to check whether the search engine is the culprit. If the number of stalls is high enough, and corr_fifo stalls are comparable, then that scenario is a good candidate to increase the nand_speed_clk frequency.

Now the duration for accumulation is left entirely to software. It is the software's task to manage and reset these registers appropriately. This is done purposely so software can decide over what granularity it wants to make the perf/watt decision. After software has read out this value and has no more use for it, it should write 32'h0 into the two register to reset their value, for the next accumulation as required. But it can choose accumulate the statistics over several DMA transfers until the max value of 32'hffffffff has reached. Hardware guarantees that the accumulation of stall cycles will not overflow.

29.3.10 SGDMA Mode of Operation

In Tegra 3 devices, the NAND controller supports an indirect scheme of addressing memory on the AHB bus. Software sets up descriptors in address length pairs to describe each fragment of memory, the set of which constitutes the data that needs to be written to/read from the flash device.

This section describes the DMA Mechanism from a software viewpoint (It concentrates on the SG-DMA mode only):

29.3.10.1 SG-DMA Mode Registers

Registers	Relevant Fields	Notes
NAND_DMA_MST_C TRL_0	DMA_GO, DIR, BURST_SIZE, DMA_EN_A, DMA_EN_B	Most of the sense of these fields is similar in SG-DMA mode as in normal mode.
NAND_DMA_CFG_A_0	DMA_BLOCK_SIZE_A	In the normal this describes DATA size in bytes. In SGDMA mode, this denotes no. of address length pairs in the DATA descriptor table. . In both modes, n-1 value is adhered to maintain compatibility.
NAND_DMA_CFG_B_0	DMA_BLOCK_SIZE_B	In the normal this describes TAG size in bytes. In SGDMA mode, this denotes no. of address length pairs in the TAG descriptor table. In both modes, n-1 value is adhered to maintain compatibility.
NAND_DATA_BLOCK_PTR_0		In the normal mode this points to start of DMA data block. In SGDMA mode, this points to start address of DATA descriptor table.
NAND_TAG_BLOCK_PTR_0		In the normal mode this points to start of DMA data block. In SGDMA mode, this points to start address of TAG descriptor table.
NAND_CONFIG2_0	SGDMA_EN	Single bit selector that tells hardware to operate in SGDMA mode.

Example

Suppose software needs to transfer 64kB of memory to the flash. (Flash write operation).

The OS may have this data in several data fragments in system memory. As opposed to forming the contiguous memory set in chips before Tegra 2, software now needs to set up the descriptor table, with the start address and length (bytes) of each fragment. The DMA engine will automatically fetch all the data, align all the fetches.

The following example shows a descriptor table. (Main)

At the starting address pointed by NAND_DATA_BLOCK_PTR_0 with following offsets.

OFFSET 0x00	Address 1 [31:0]
0x04	Length1 [15:0] padded with 16 0-msbs
0x08	Address2[31:0]
0x0c	Length2 [15:0] padded with 16 0-msbs
-----	---------
0x30	Address7 [31:0]
0x34	Length7 [15:0] padded with 16 0-msbs

For this table, the register programming would be:

- NAND_DMA_MST_C TRL_0 — Fields have same meaning as in normal dma case
- NAND_DMA_CFG_A_0 — 6 (7 -1)
- NAND_DMA_CFG_B_0 — NA
- NAND_DATA_BLOCK_PTR_0 — <start addr>
- NAND_TAG_BLOCK_PTR_0 — NA
- NAND_CONFIG2_0 — SGDMA_EN=1

29.3.10.2 Programming Notes

- Descriptors have complete flexibility to describe anything from 1 byte to an entire transfer.
- Descriptor lengths need not be word or half word aligned i.e. byte granular lengths are permitted. It follows that descriptor addresses can also be byte aligned.
- Descriptor need not align to sector boundaries.
- *Descriptors need not align to page boundaries. The DMA automatically context switches at a page boundary, saves the current fetch context and restores it at the next main/tag page.
- The data burst size is same as the programmed data burst. However, depending on the alignment of descriptors, the controller reduces the burst in cases when there is a need to align to page boundaries, descriptor boundaries and start addresses of the fetch. The burst size, however, is guaranteed to be equal to or less than the programmed size.
- Descriptor entry fetch is always done in single word bursts.
- Main and Tag areas have separate descriptor tables each, each describing the respective data areas.
- The sum of lengths of the descriptor entries is the intended data transfer length. This is NOT explicitly programmed in SG-DMA mode. Only the table and the number of addr-length pairs in the table (table size) are programmed. Each address and length is 32-bit fields (length is actually 16 bit but we align it to a word boundary).

29.3.10.3 Hardware prefetch/ performance Notes

Hardware does not sequentially fetch descriptor and respective data. The tables are prefetched ahead of time, and also dynamically filled (dependent on data arbitration and bubbles in the data fetching) so that the next data fetch address is

(almost) always ready on the completion of a descriptor data fragment. Hence we always fetch descriptor in single burst to avoid descriptor to clog the fetch. Main and Tag descriptors are filled based on the occupancy levels of the respective buffers. This mechanism will prevent a performance drop when the data is very fragmented i.e. size of descriptor lengths is small, and no. of descriptors is large.

29.3.11 Abort Operation

Tegra 3 NAND supports an alternative to module reset for a prospective clean exit when software decides to abort an operation. Functionally, an abort is supported at any time after a valid transaction has been started, but typically would only abort on the following cases. Each of the below cases represents an unexpected error case, and causes an interrupt, upon which software would typically abort the transaction.

29.3.11.1 Triggers to an Abort Condition

- Lock Error
- Decode fatal error
- Rd status error (cmdQ error)

29.3.11.2 Abort Programming Sequence

On receiving any of the above triggers (or otherwise), software can set the abort bit in CONFIG2 register (bit 31) to initiate an abort. The Controller then goes through a series of operations depending on when and in what operation the abort was received, and sets the abort_done bit in the ISR after a few clock cycles and generates an interrupt with the abort_done bit set in the ISR. At that point, software is expected to reprogram all state and config registers in the next transaction like a brand new transaction. In Tegra 3, a “resume” is NOT supported.

It is essential while aborting a non-ecc use case i.e. when config bch_ecc bit = 0, that the nand_speed_clk is shut off. This should be the case to save power anyway, but software must sure to turn off nand_speed_clk.

29.3.12 NAND Timing Registers

Hardware implementation of timings in NAND controller follows the simple implementation of running timing counter for no. of clocks for programmed. Timing equations representing the timing parameter implementation are described below. For different Flash timings or for different NAND Clock frequency options timing parameters be calculated and programmed in respective timing registers. Following equation cites an example of such calculation:

Consider AC timing parameter tWP from flash requirement is 20ns. Governing equation for this timing parameter is,

tWP (in ns) = (TWP count programmed + 1) X NAND_CLK_PERIOD

So required bit field programming value of TWP_CNT = tWP(ns)/NAND_CLK_PERIOD -1

Considering NAND Clock source is selected for 100MHz frequency, resulting timing count to be programmed is,

TWP_CNT = 20ns/10ns (clock period) -1 = 1

Table 78 ASYNC Mode Timing Equations with TIMING_SEL=0 (NAND_TIMING, NAND_TIMING2)

TIMING_VALUE	Guideline	Timing equation
TWP_CNT	Refer to tWP timing from flash datasheet	tWP = [TWP_CNT +1] * NAND_CLK_PERIOD
TWH_CNT	Refer to tWH timing from flash datasheet	tWH = [TWH_CNT +1] * NAND_CLK_PERIOD
TRP_CNT	Non-EDO mode: Max(tRP, tREA) timings from flash datasheet + 6ns (round trip delay) EDO mode: tRP timing from flash datasheet	Max(tRP,tREA) + Round_Trip_Delay = [TRP_CNT +1] * NAND_CLK_PERIOD tRP = [TRP_CNT +1] * NAND_CLK_PERIOD

TIMING_VALUE	Guideline	Timing equation
TRH_CNT	Refer to tRH/tREH timing from flash datasheet	tRH = [TRH_CNT +1] * NAND_CLK_PERIOD
TCS_CNT	Program for Max(tCS, tCH, tALS, tALH, tCLS, TCLH) timings from flash datasheet	Max(tCS, tCH, tALS, tALH, tCLS, TCLH) = [TCS_CNT + TWP_CNT +2] * NAND_CLK_PERIOD
TCR_TAR_TRR_CNT	Program Max(tCR, tAR, tRR) timings from flash data sheet	Max(tCR, tAR, tRR) = (TCR_TAR_TRR_CNT+4) * NAND_CLK_PERIOD
TWHR_CNT	Program Max(tWHR,tWHR2) timing from flash data sheet	Max(tWHR,tWHR2) = (TWHR_CNT+1) * NAND_CLK_PERIOD
TWB_CNT	Refer to tWB timing from flash datasheet	tWB = (TWB _CNT+ TWH_CNT + TCS_CNT - 6) * NAND_CLK_PERIOD
TADL_CNT	Refer to tADL timing from flash datasheet	tADL = (TADL_CNT + TWH_CNT + TWP_CNT + 6) * NAND_CLK_PERIOD

Table 79 ASYNC Mode Timing Equations with TIMING_SEL=1(NAND_ASYNC_TIMING_0-3)

TIMING_VALUE	Guideline	Timing equation
TWP_CNT	Refer to tWP timing from flash datasheet	tWP = [TWP_CNT +1] * NAND_CLK_PERIOD
TWH_CNT	Refer to tWH timing from flash datasheet	tWH = [TWH_CNT +1] * NAND_CLK_PERIOD
TRP_CNT	Non-EDO mode: Max(tRP, tREA) timings from flash datasheet + 6ns (round trip delay) EDO mode: tRP timing from flash datasheet	Generated timing for ASYNC modes, Max(tRP,tREA) + 6ns = [TRP_CNT +1] * NAND_CLK_PERIOD tRP = [TRP_CNT +1] * NAND_CLK_PERIOD Fixed generated timing in DDR modes, tRP = (NAND_CLK_PERIOD/2) ns. Do not impact with TRP_CNT programmed value for all DDR modes.
TREA_CNT	Non-EDO mode: Max(tRP, tREA) timings from flash datasheet + 6ns (round trip delay) EDO mode: tRP timing from flash datasheet	Max(tRP,tREA) + 6ns = [TREA_CNT +1] * NAND_CLK_PERIOD tRP = [TREA_CNT +1] * NAND_CLK_PERIOD Not required for to program TREA_CNT for DDR all modes.
TRH_CNT	Refer to tRH/tREH timing from flash datasheet	tRH = [TRH_CNT +1] * NAND_CLK_PERIOD. Fixed generated timing in toggle DDR mode, tREH = (NAND_CLK_PERIOD/2)ns Fixed generated timing in DDR modes, tRP = (NAND_CLK_PERIOD/2) ns. Do not impact with TRH_CNT programmed value for all DDR modes.
TCLH_CNT	Max(tCH,tCLH) timings from flash datasheet	Generated timing for all ASYNC and toggle DDR, tCLH = (TCLH_CNT + TWH_CNT + 3) * NAND_CLK_PERIOD ns Fixed generated timing for ONFI DDR, tCLH = NAND_CLK_PERIOD/2 ns. Do not impact with TCLH_CNT programmed value for ONFI DDR mode. For new Samsung devices with tCWAW timing requirement, tCLH = (TCWAW_CNT + TWH_CNT + TALS_CNT + TWP_CNT + 5) * NAND_CLK_PERIOD
TCH_CNT	Reserved bit field. Hardware implementation of tCLH timing and other timing parameter guarantees this timing requirement.	
TALH_CNT	Refer to tALH timing from flash datasheet	Generated timing for all ASYNC and toggle DDR, tALH = (TALH_CNT + TWH_CNT + 3) * NAND_CLK_PERIOD

TIMING_VALUE	Guideline	Timing equation
		Fixed generated timing for ONFI DDR, tALH = NAND_CLK_PERIOD/2 ns. Do not impact with TALH_CNT programmed value for ONFI DDR mode.
TCLS_CNT	Refer to tCLS timing from flash datasheet	Generated timing for all ASYNC and toggle DDR, tCLS = (TCLS_CNT + TWP_CNT + 2) * NAND_CLK_PERIOD Fixed generated timing for ONFI DDR, tCLS = NAND_CLK_PERIOD/2 ns. Do not impact with TCLS_CNT programmed value for ONFI DDR mode.
TALS_CNT	Refer to tALS timing from flash datasheet	Generated timing for all ASYNC and toggle DDR, tALS = (TALS_CNT + TWP_CNT + 2) * NAND_CLK_PERIOD Fixed generated timing for ONFI DDR, tALS = NAND_CLK_PERIOD/2 ns. Do not impact with TALS_CNT programmed value for ONFI DDR mode.
TCS_CNT	Refer to tCS timing from flash datasheet	Generated timing for all ASYNC and toggle DDR, tCS = (TCS_CNT + TCLS_CNT + TWP_CNT + 4) * NAND_CLK_PERIOD ns Generated timing for ONFI DDR, tCS = (TCS_CNT + 1) * NAND_CLK_PERIOD ns If Flash is not ready busy and controller waiting for RBSY_CHK condition, that time will add up.
TCR_CNT	Reserved bit field. Programming proper TRR_CNT meets this timing	
TAR_CNT	Refer to tAR, tWHR timings from flash datasheet	Address cycle ALE low to data output time Covered by tRR timing during Data read operation. Generated timing during Read ID, Max(tAR, tWHR) = (TAR_CNT+1) * NAND_CLK_PERIOD ns This timing parameter is not relevant for ONFI DDR mode. Do not impact with TAR_CNT programmed value for ONFI DDR mode.
TRR_CNT	Refer to tRR timing from flash datasheet	tRR = (TRR_CNT + 4) * NAND_CLK_PERIOD
TWHR_CNT	Refer to tWHR timing from flash datasheet	Generated timing for async and Toggle DDR modes, tWHR = (TWH_CNT + TCLH_CNT + TWHR_CNT + TRR_CNT + 5) * NAND_CLK_PERIOD ns. Generated timing for ONFI DDR modes, tWHR = (TCAD_CNT + TWHR_CNT + 4) * NAND_CLK_PERIOD ns.
TWHR2_CNT	Refer to tWHR2 timing from flash datasheet	Relevant for Samsung new devices only, Generated timing, tWHR2 = (TCLH_CNT + TWH_CNT + TWHR2_CNT + 6) * NAND_CLK_PERIOD ns This timing parameter is not relevant for ONFI DDR mode. Do not impact with TWHR2_CNT programmed value for ONFI DDR mode.
TWB_CNT	Refer to tWB timing from flash datasheet	Generated timing for all async and toggle DDR modes, tWB = (TWH_CNT + TCLH_CNT + TWB_CNT - 6) * NAND_CLK_PERIOD ns. Generated timing for ONFI DDR, tWB = (TCAD_CNT + TWB_CNT - 6) * NAND_CLK_PERIOD ns.
TADL_CNT	Refer to tADL timing from flash datasheet	Generated timing for ASYNC and toggle DDR, tADL = (TWH_CNT + TALH_CNT + TADL_CNT + 4) * NAND_CLK_PERIOD ns Generated timing for ONFI DDR, tADL = (TCAD_CNT + TADL_CNT + 4) *

TIMING_VALUE	Guideline	Timing equation
		NAND_CLK_PERIOD ns
TRHW_CNT	Refer to tRHW timing from flash datasheet	Generated timing for all async and toggle DDR modes, tRHW = (TRHW_CNT + TREH_CNT + TCLS_CNT + 4) * NAND_CLK_PERIOD ns. For ONFI DDR modes, tRHW = (TRHW_CNT + 1) * NAND_CLK_PERIOD ns.
TCWAW_CNT	Refer to tCWAW timing from flash datasheet	Relevant for Samsung new devices only. Generated timing for async and toggle DDR modes, tCWAW = (TCWAW_CNT + TWH_CNT + TALS_CNT + TWP_CNT + 5) * NAND_CLK_PERIOD ns This timing parameter is not relevant for ONFI DDR mode. Do not impact with TCWAW_CNT programmed value for ONFI DDR mode.

Table 80 NAND Toggle DDR Timing equations (NAND_TDDR_TIMING0-1)

TIMING_VALUE	Guideline	Timing equation
TRPSTH_CNT	Refer to tRPSTH timing from flash datasheet	tRPSTH = [TRPSTH_CNT +1] * NAND_CLK_PERIOD
TRPST_CNT	Refer to tRPST timing from flash datasheet	tRPST = [TRPST_CNT +2] * NAND_CLK_PERIOD
TRPRE_CNT	Refer to tRPRE timing from flash datasheet	tRPRE = [TRPRE_CNT +1] * NAND_CLK_PERIOD
TCDQSH_CNT	Refer to tCDQSH timing from flash datasheet	tCDQSH = (TWH_CNT + TCLH_CNT +TCDQSH_CNT +5) * NAND_CLK_PERIOD
TCDQSS_CNT	Refer to tCDQSS timing from flash datasheet	tCDQSS = TCDQSS_CNT * NAND_CLK_PERIOD
TWPSTH_CNT	Refer to tWPSTH timing from flash datasheet	tWPSTH = (TWPSTH_CNT + 1) * NAND_CLK_PERIOD
TWPST_CNT	Refer to tWPST timing from flash datasheet	tWPST = (TWPST_CNT + 4) * NAND_CLK_PERIOD
TWPRE_CNT	Refer to tWPRE timing from flash datasheet	tWPRE = (TCDQSS_CNT + TADL_CNT + TWPRE_CNT + 2) * NAND_CLK_PERIOD

Table 81 NAND SDDR timing equations (NAND_SDDR_TIMING_0-1)

TIMING_VALUE	Guideline	Timing equation
TCKWR_CNT	Refer to tCKWR timing from flash	Generated timing, tCKWR = (TCKWR_CNT +1) * NAND_CLK_PERIOD ns
TCAD_CNT	Refer to tCAD timing from flash	Command, Address, Data delay Generated timing, tCAD = (TCAD_CNT+1) * NAND_CLK_PERIOD ns.
TWRCK_CNT	Refer to tWRCK timing from flash	Generated timing tWRCK = (TWRCK_CNT +1) * NAND_CLK_PERIOD ns.
TWPST_CNT	Refer to tWPST timing from flash	Generated timing, tWPST = 3 * NAND_CLK Required programmed values for all clock frequencies or devices TWPST_CNT = 0.
TWPRE_CNT	Refer to tWPRE timing from flash	Generated timing tDQSS = 1 * NAND_CLK Generated timing tWRPRE = 3 * NAND_CLK. Generated timing tDQSS = 1 * NAND_CLK. Required programmed values for all clock frequencies or devices TWPRE_CNT = 0.
TCCS_CNT	Refer to tCCS timing from flash	Generated timing tCCS = (TCCS_CNT + TCAD_CNT + 4) * NAND_CLK_PERIOD ns.

29.3.13 Round Trip Delay Calculation for tRP Timing

Standard asynchronous interface for read operation require timing adjustment so that NAND controller properly samples the read data coming in from flash. This is because NAND controller samples the read data coming from flash with the same synchronous clock (NAND_CLK) used for read enable (REN_) signal generation at the end of Read pulse timing. Because of this implementation approach, delay incurred in REN signal going, and delay incurred in DATA coming including on board should be accounted for. tRP timing calculation. Hence Read Pulse width timing has to be extended by tDLY timing to account for round trip delay.

Total delay tDLY = REN out delay + REN out delay on board + DATA out delay from flash to chip input + DATA INPUT pad delay + setup time of flop

Essentially above equation is simplified to,

tDLY = REN out PAD delay + Round trip delay on board + Max(DATA INPUT pad delay) + setup time of flop

Where Round trip delay on board = Max (REN out delay on board + DATA out delay from flash to chip input) based on board design.

Based on Tegra 3 silicon timing data, the above equation reduces to:

tDLY = 3.3 ns + Round trip delay on board + 2ns

= 5.3 ns + Round trip delay on board (<500ps typically)

29.3.14 Read/Write Cycle Time Programming

It is common that from flash timing requirements, total cycle time of read/write access is slightly more than pulse width time (tRP/tWP) plus hold time (tREH/tWH). tRP, tREH achieved timings shall make sure that tRC timing requirement is also met with whatever NAND clock selection. For example, for one of the Samsung flash timing requirement are such as:

tRP=12, tREH=10 but tRC = 25ns (tRP + tREH < tRC)

From achieved timing requirements point of view, all these 3 should be satisfied not just tRP/tREH alone. In NAND controller tRP/tRP_RESP, tREH, tWP, tWH timings are programmable. By virtue of generated timings of these individual timing parameters, tRC, tWC timings can be met.

Achieved Read cycle time (tRC) is nothing but,

Read pulse width time + Read Hold time => tRP/TRP_RESP timing + tREH timing

Achieved Write cycle time (tWC) is nothing but,

Write pulse width time + Write hold time ==> tWP timing + tWH timing

For the same illustration described above, adjustment of 3ns to satisfy read/write cycle timing can be achieved:

Option1: tRP = 15ns, tREH=10ns

Option2: tRP = 12ns, tREH=13ns

Option2 is preferable, because with NAND_CLK_PERIOD=12.5ns all these timing requirements are met closely with slight margin. Also to achieve option1 timings, NAND_CLK resolution of 5ns period is required which is costly from power front. So when choosing tRP, tREH timing requirements, total cycle timing requirements have to be taken into account.

29.3.15 tRP Programming for EDO Mode of Operation

In EDO mode of operation NAND controller samples read data on completion of Read Cycle timing. With this effect for tRP calculation tDLY adjustment as described in earlier section is not required provided, tDLY < tRC condition is met. If tDLY is more than tRC, the delay adjustment required for EDO mode will be,

tDLY_EDO = (tDLY + Round trip board delay) – tRC time

Also the requirement for tRP timing of, Max (tRP, tREA) will also be gone since tREA is also less than tRC practically for any flash. So tRP timing considering the total Read Cycle timing requirement as described in Read/Write cycles programming section will suffice, and there is no need to extend tRP pulse width. One can observe 30% better performance improvement using EDO mode compared to non-EDO mode of operation.

29.3.16 QUSE/DQS Delay Programming for DDR Modes

Proper QUSE/DQS programming is required to sample incoming read data in DDR modes. Following guideline should be maintained along with characterization to reliably work with all DDR variants of devices.

The programming guide line for following registers ignoring the PVT variation and based on Silicon validation platform is given below.

FBIO_CFG.CFG_QUSE_LATE, FBIO_QUSE_DLY and CFG_DQSIB_DLY

- QUSE coarse delay setting based on NAND_CLK,

ONFI/Toggle DDR modes

Frequency	QUSE_LATE setting
0 < Freq < 24MHz	FBIO_CFG.CFG_QUSE_LATE [3:0] = 4'b0
24 <= Freq < 66MHz	FBIO_CFG.CFG_QUSE_LATE [1] =1
66 <= Freq < 100MHz	FBIO_CFG.CFG_QUSE_LATE [2] =1

- QUSE fine tune delay from delay line programming, FBIO_QUSE_DLY. CFG_QUSE_DLY_BYTE_0 = 0x00
- DQS fine tune delay from delay line programming such as,

  NAND_FBIO_DQSIB_DLY_0 = ((1.25 * (1000/NAND_CLK frequency))>=63)? 63: (1.25 * (TIMESCALE/NAND_CLK frequency));

29.3.17 Safe Timing Values and Trimmer Settings for Boot ROM

Purpose of this section is to provide guidance on safe value settings for timing parameters as well as QUSE/DQS delay values during boot. Since ONFI compliant DDR devices can boot up in regular ASYNC mode, it's not recommend to operate to turn on DDR modes during boot for such flashes. Any required trimmer/timing register settings can be programmed through BCT to support switching to DDR mode follow-on to boot. Initialization and command sequences of ONFI standard mode should be adopted in bootrom driver like in async mode of operation.

During boot, as the clocking runs on at crystal frequency of 24MHz and hardware ensures by design that it's possible to come up with safe settings of various timing register and programmable trimmer values for reliable working of various type of flash devices for Async interface and Toggle Interface. Since Toggle devices data transfer is DDR only requires the boot support and hence Safe value programming.

Below references of register settings for timing and trimmer settings cover legacy Async interface/ONFI Async interface and Toggle DDR interface. In summary -

- Legacy Async interface or Samsung devices including toggle DDR flash.
  - ASYNC only flashes controller is remains in default ASYNC mode of operation.
  - After Toggle flash identification, controller switched to toggle DDR mode of operation.
- ONFI devices including DDR devices only in ASYNC mode.
  - After device initialization and identification sequences remains in default ASYNC mode of operation.

Safe Timing register values are,

Timing Register	Required register setting
NAND_ASYNC_TIMING0	0x0A00_0A0A
NAND_ASYNC_TIMING1	0x050A_010A
NAND_ASYNC_TIMING2	0x0A00_0000
NAND_ASYNC_TIMING3	0x0001_1111
NAND_TDDR_TIMING0	0x0000_0111
NAND_TDDR_TIMING1	0x0004_0110

Safe QUSE/DQS delay settings for toggle DDR support are,

Register	Required field	Value
FBIO_CFG	CFG_QUSE_LATE	0x0
FBIO_QUSE_DLY	CFG_QUSE_DLY_BYTE_0	0x0
FBIO_DQSIB_DLY	CFG_DQSIB_DLY_BYTE_0	0x3F

29.3.17.1 Command Queue Abort/Stall Conditions

When there is failure in any previous command sequence operation, Command queue execution will be aborted triggering interrupt to processor, if RD_STATUS_CHK is enabled. It is possible that,

- Current Register status will not reflect the register configured which caused the failure.
- SW has to figure out what is the last command sequence to the flash which caused this failure.
- Re arranges the list as per application, and configures the Command queue configuration registers to re-start new sequence of operations.

If RD_STATUS_CHK fails for any of the command packet execution is stopped right away, i.e. at this time the command packet which needs the RD_STAUS_CHK is read from the buffer and executed up to the RD_STATUS_CHK only. Associated read/write/command transaction to flash as programmed NAND_COMMAND register is abandoned further and software has to come back from this packet for recovery procedure.

Also software can halt the command queue execution at any point of time by clearing the LL_START bit of NAND_LL_CONFIG register. Removing LL_START is allowed at any time during command queue execution and is interpreted as a way of disabling the command queue execution. By this time recognizing the LL_START clear, NAND controller should have extracted next packet command and accordingly NAND_LL_STATUS registers will reflect the status as in Abort condition.

Word counter status registers in NAND_LL_STATUS register will be useful for calculating the next start address for both cases. However, in case of Halt, please note that enabling the LL_START will not resume execution and will repeat the execution from start. Software should configure NAND_LL_PTR register for necessary correction and can continue execution enabling LL_START.

Next start address after Abort/Halt should be calculated as:

> *NAND_LL_PTR + (LL_LENGTH_DONE – LL_LENGHT_LAST_PACKET) * 4 = Byte Address of the command packet where the command queue execution stopped either because of the COMMAND QUEUE Error [RD_STATUS_CHK error] or COMMAND QUEUE Stall.

29.4 NAND Controller Registers

29.4.1 NAND_COMMAND_0

Configuration of this register determines the flash sequence to be initiated by NAND controller.

NAND Command Register

Offset: 000h | Read/Write: R/W | Reset: 0b00000000100000000000000000000100

Bit	Reset	Description
31	0x0	GO: 0 = HW clears when programmed nand IO operation is completed. 0 = DISABLE 1 = ENABLE
30	0x0	CLE: CLE enable 1 = Flash sequence has Command Cycle(CLE) enabled 0 = Flash sequence has Command Cycle(CLE) disabled 0 = DISABLE 1 = ENABLE
29	0x0	ALE: ALE enable 1 = Flash sequence has Address Cycle(CLE) enabled 0 = Flash sequence has Address Cycle(CLE) disabled 0 = DISABLE 1 = ENABLE
28	0x0	PIOPIO mode of operation enable 1 = Dataout is from NAND_RESP register and Datain is to NAND_RESP register 0 = Dataout is from FIFO buffer and Datain to FIFO buffer 0 = DISABLE 1 = ENABLE
27	0x0	TX: write data transfer enable - required for FLASH program 1 = Write data transfers to flash is enabled 0 = Write data transfers to flash is disabled. 0 = DISABLE 1 = ENABLE
26	0x0	RX: read data transfer enabled - required for FLASH read 1 = Read data transfers from flash is enabled 0 = Read data transfers from flash is disabled. 0 = DISABLE 1 = ENABLE
25	0x0	SEC_CMD: CMD2 sequence to flash enable 1 = NAND command sequence have a second command(CLE) cycle 0 = NAND command sequence doesn't have second CLE cycle 0 = DISABLE 1 = ENABLE
24	0x0	AFT_DAT: CMD2 placement control 1 - CMD2 CLE cycle is issued after data transfer cycles. this is the typical usage during FLASH program 0 - CMD2 CLE cycle is issued right after Address transfer cycles, typical usage during FLASH read 0 = DISABLE 1 = ENABLE
23:20	0x8	TRANS_SIZE: Transfer size of bytes depends on PAGE_SIZE_SEL field of NAND_CONFIG register 0 = BYTES1 1 = BYTES2 2 = BYTES3 3 = BYTES4 4 = BYTES5 5 = BYTES6 6 = BYTES7 7 = BYTES8 8 = BYTES_PAGE_SIZE_SEL

Bit	Reset	Description
19	0x0	A_VALID: Main data region transfer enable 1 = Involves Main area data transfer in flash sequence 0 = Doesn't involve Main area data transfer 0 = DISABLE 1 = ENABLE
18	0x0	B_VALID: Spare region (aka TAG) transfer enable 1 = Involves spare area data transfer in flash sequence 0 = Doesn't involve spare area data transfer 0 = DISABLE 1 = ENABLE
17	0x0	RD_STATUS_CHK: Hardware assisted read status check enable 1 = Indicates to controller that current IO sequence need RD STATUS check condition to be qualified. 0 = auto read status check is disabled notes: please refer to NAND_HWSTATUS_CMD register for qualifier condition 0 = DISABLE 1 = ENABLE
16	0x0	RBSY_CHK: Hardware assisted rbsy check enable 1 = Indicates to controller that current IO sequence need RBSY check condition to be qualified. 0 = auto RBSY check is disabled notes: please refer to NAND_HWSTATUS_CMD register for qualifier condition 0 = DISABLE 1 = ENABLE
15	0x0	CE7: Chip select enable for Flash Card 7 0 = DISABLE 1 = ENABLE
14	0x0	CE6: Chip select enable for Flash Card 6 0 = DISABLE 1 = ENABLE
13	0x0	CE5: Chip select enable for Flash Card 5 0 = DISABLE 1 = ENABLE
12	0x0	CE4: Chip select enable for Flash Card 4 0 = DISABLE 1 = ENABLE
11	0x0	CE3: Chip select enable for Flash Card 3 0 = DISABLE 1 = ENABLE
10	0x0	CE2: Chip select enable for Flash Card 2 0 = DISABLE 1 = ENABLE
9	0x0	CE1: Chip select enable for Flash Card 1 0 = DISABLE 1 = ENABLE
8	0x0	CE0: Chip select enable for Flash Card 0 0 = DISABLE 1 = ENABLE
7	0x0	RD_STATUS_END: Do page read status command sequence after data read 0 = DISABLE 1 = ENABLE
6	0x0	WAIT_RBSY_LOW: Wait for RBSY low before deactivating the chip select to complete the flash sequence 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
5:4	0x0	CLE_BYTE_SIZE: Command cycle byte count 0 = CLE_BYTES1 1 = CLE_BYTES2 2 = CLE_BYTES3 3 = CLE_BYTES4
3:0	0x4	ALE_BYTE_SIZE: Address cycle byte count 0 = ALE_BYTES1 1 = ALE_BYTES2 2 = ALE_BYTES3 3 = ALE_BYTES4 4 = ALE_BYTES5 5 = ALE_BYTES6 6 = ALE_BYTES7 7 = ALE_BYTES8 8 = ALE_BYTES9 : Reserved 9 = ALE_BYTES10 : Reserved 10 = ALE_BYTES11 : Reserved 11 = ALE_BYTES12 : Reserved 12 = ALE_BYTES13 : Reserved 13 = ALE_BYTES14 : Reserved 14 = ALE_BYTES15 : Reserved 15 = ALE_BYTES16

29.4.2 NAND_STATUS_0

Collection of NAND flash RBSY line status and other internal operations status.

NAND Status Register

Offset: 004h | Read/Write: RO | Reset: 0bxxxxxxxx00xxxxxx1

Bit	Reset	Description
15	X	RBSY7: 1 = Indicates flash7 is RDY
14	X	RBSY6: 1 = Indicates flash6 is RDY
13	X	RBSY5: 1 = Indicates flash5 is RDY
12	X	RBSY4: 1 = Indicates flash4 is RDY
11	X	RBSY3: 1 = Indicates flash3 is RDY
10	X	RBSY2: 1 = Indicates flash2 is RDY
9	X	RBSY1: 1 = Indicates flash1 is RDY
8	X	RBSY0: 1 = Indicates flash0 is RDY
7	0	WR_ACT: 1 = Indicates write cycles to flash are in progress
6	0	RD_ACT: 1 = Indicates read cycles to flash are in progress
0	1	ISEMPTY: 1 = Indicates NAND controller is in IDLE state of operation, and there are no flash/DMA transactions pending.

29.4.3 NAND_ISR_0

Interrupt status hardware behavior: All interrupt status bits are SET, respective event occurs and writes 1 to clear.

NAND Interrupt Status Register

Offset: 008h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxxxxxx00000000000000

Bit	R/W	Reset	Description
31	RW	0x0	ABORT_DONE: 1 = This interrupt is generated after an abort was initiated by software by setting CONFIG2_ABORT bit, and hardware has finished abort sequence and returned to Idle ready for next flash operation. Note that it is required of software to reprogram all state/registers/pointers like a new transaction. (Hardware will typically hold data registers as is. and reset control registers after an abort)
24	RO	X	CORRFAIL_ERR: 1 = Correctable OR Un-correctable errors occurred in the DMA transfer without regard to HW_ERR_CORRECTION feature is enabled or not. Use extended decode results in NAND_DEC_RESULT and NAND_DEC_STATUS_EXT to figure out further action for block replacement/wear leveling during file system management for software. Covers all ECC selection: RS/Hamming/BCH modes
15	RW	0x0	IS_RBSY7: 1 = Flash7 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
14	RW	0x0	IS_RBSY6: 1 = Flash6 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
13	RW	0x0	IS_RBSY5: 1 = Flash5 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
12	RW	0x0	IS_RBSY4: 1 = Flash4 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
11	RW	0x0	IS_RBSY3: 1 = Flash3 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
10	RW	0x0	IS_RBSY2: 1 = Flash2 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
9	RW	0x0	IS_RBSY1: 1 = Flash1 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
8	RW	0x0	IS_RBSY0: 1 = Flash0 is Ready interrupt occurred. This bit is set only when not running in COMMAND QUEUE MODE
7	RW	0x0	IS_UND: 1 = FIFO under run interrupt occurred this should not happen in general usage, if it happens there is a potential hardware issue.
6	RW	0x0	IS_OVR: 1 = FIFO is Overrun this should not happen in general usage, if it happens there is potential hardware issue.
5	RW	0x0	IS_CMD_DONE: 1 = Command operations are completed as per NAND command register settings. This is set ONLY when not running in COMMAND QUEUE MODE
4	RW	0x0	IS_ECC_ERR: 1 = ECC error generated for following reasons. ->ecc decode resulted in uncorrectable errors in one of sector(sub-page) ->ecc decode resulted in correctable errors more than trigger level as defined in TRIG_LVL in NAND_CONFIG register. Bit is set for legacy mode of ECC selection with HW_ECC & ECC_TAG_EN only i.e. for RS/Hamming selection. This bit is not set for BCH selection
3	RW	0x0	IS_LL_DONE: 1 = Command queue execution completed
2	RW	0x0	IS_LL_ERR: 1 = One of the Command queue packet execution returned error because of Read Status Check failure

29.4.4 NAND_IER_0

NAND Interrupt Enable Register

Offset: 00ch | Read/Write: R/W | Reset: 0b0xxxxxxxxxxx00000000000000000x0

Bit	Reset	Description
31	0x0	ABORT_DONE: 1 = This interrupt is generated after an abort was initiated by software by setting CONFIG2_ABORT bit, and hardware has finished abort sequence and returned to Idle ready for next flash operation. Note that it is required of software to reprogram all state/registers/pointers like a new transaction. (Hardware will typically hold data registers as is. and reset control registers after an abort)
19:16	0x0	ERR_TRIG_VAL: Trigger for correctable error Interrupts by main ECC RS decoder, if HW_ERR_CORRECTION feature is enabled. Mechanism for software to get an idea on error pattern development over a period of usage. NAND controller will trigger interrupt if the current main page read transfer resulted in correctable errors reached this trigger value for Reed-Solomon selection. For example, of ECC_ERROR interrupt for t=4, with ERR_TRIG_VAL=3 could imply only one of the following. a) If DEC_FAIL = 1, one of the sub-page decode returned failure because no. of symbol errors are more than 4. b) If DEC_FAIL = 0, one of the sub-page decode returned 3 correctable errors. 0 = CORR_ERRS_0 : Reports for every single error, equivalent to ECC_ERROR interrupt without HW_ERR_CORRECTION feature. 1 = CORR_ERRS_1 2 = CORR_ERRS_2 3 = CORR_ERRS_3 4 = CORR_ERRS_4 5 = CORR_ERRS_5 6 = CORR_ERRS_6 7 = CORR_ERRS_7 8 = CORR_ERRS_8 9 = CORR_ERRS_9 10 = CORR_ERRS_10 11 = CORR_ERRS_11 12 = CORR_ERRS_12 13 = CORR_ERRS_13 14 = CORR_ERRS_14 15 = CORR_ERRS_15
15	0x0	IE_RBSY7: Flash7 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
14	0x0	IE_RBSY6: Flash6 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
13	0x0	IE_RBSY5: Flash5 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
12	0x0	IE_RBSY4: Flash4 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
11	0x0	IE_RBSY3: Flash3 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
10	0x0	IE_RBSY2: Flash2 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
9	0x0	IE_RBSY1: Flash1 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
8	0x0	IE_RBSY0: Flash0 RBSY line High interrupt enable 0 = DISABLE 1 = ENABLE
7	0x0	IE_UND: FIFO underrun interrupt enable 0 = DISABLE 1 = ENABLE
6	0x0	IE_OVR: IFO overrun interrupt enable 0 = DISABLE 1 = ENABLE
5	0x0	IE_CMD_DONE: Command operations done interrupt enable 0 = DISABLE 1 = ENABLE
4	0x0	IE_ECC_ERR: ECC error interrupt enable. Please refer to IS_ECC_ERR above for interrupt event details 0 = DISABLE 1 = ENABLE
3	0x0	IE_LL_DONE: Command queue execution completion interrupt enable 0 = DISABLE 1 = ENABLE
2	0x0	IE_LL_ERR: Flash errors in Command queue execution error interrupt enable 0 = DISABLE 1 = ENABLE
0	0x0	GIE: 0 = Global Interrupt mask enable. Masks all of the interrupts, and interrupt to signal to CPU is disabled. 0 = DISABLE 1 = ENABLE

29.4.5 NAND_CONFIG_0

Combination of NAND_COMMAND register determines the sequence of operations to be carried on. ECC selection and strength etc are selectable from this register.

NAND Configuration Register

Offset: 010h | Read/Write: R/W | Reset: 0b00010x00000000110000000000000000

Bit	Reset	Description
31	0x0	HW_ECC: Hardware Error detection enable for Main page read data 0 = DISABLE 1 = ENABLE
30	0x0	ECC_SEL: Hardware Error detection/correction algorithm selection 0 = HAMMING 1 = RS
29	0x0	HW_ERR_CORRECTION: Enable Auto hardware error correction. Emulates software behavior of reading the error vectors from system buffer as pointed in the error vector address register, applies correction and updates the memory word with corrected data. This is done on page basis as soon as the decode information is available as the flash read is placed in memory. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
28	0x1	PIPELINE_EN: Pipelines of decode operation enable. Enable next page flash READ data transfer even before current page ECC Decode is completed. If disabled, new page READ is started only after the previous page flash read, ECC decode(detection) are completed. 0 = DISABLE 1 = ENABLE
27	0x0	ECC_EN_TAG: Hardware Error detection enable for Spare read data 0 = DISABLE 1 = ENABLE
25:24	0x0	TVALUE: Hardware Error correction algorithm strength selection for RS ECC selection. 0 = TVAL4 : (t=4) 4 bit error correction per each 512 bytes of data 1 = TVAL6 : (t=6) 6 bit error correction per each 512 bytes of data 2 = TVAL8 : (t=8) 8 bit error correction per each 512 bytes of data 3 = TVAL_RSVD
23	0x0	SKIP_SPARE: Skip spare feature enable. SKIP_SPAE_SEL below indicates how many bytes in spare area of flash to be skipped over either for reading/writing. 0 = DISABLE 1 = ENABLE
22	0x0	COM_BSY: Combined RBSY selection enabled. 0 = DISABLE: RBSY0 is from Flash card 0 1 = ENABLE: RBSY0 seen by hardware is wired AND of all flash cards connected
21	0x0	BUS_WIDTH: Flash read/write data bus width 0 = BUS_WIDTH_8 : selection Data bus width 8-bit 1 = BUS_WIDTH_16 : selection Data bus width 16-bit
20	0x0	LPDDR1_MODE: LPDDR1 mode of pin ordering enable. Pin ordering to be package friendly with LPDDR1 0 = DISABLE : Standard mode of pin ordering 1 = ENABLE : Pin ordering compatible with LP-DDR1 multi chip package
19	0x0	EDO_MODE: EDO mode of flash read enable. 0 = DISABLE : Data sampled on positive edge of REN 1 = ENABLE : Data sampled on completion of read cycle time
18:16	0x3	PAGE_SIZE_SEL: Page size selection - depends on Flash memory organization used. 0 = PAGE_SIZE_256 : 256 bytes size is not supported - RSVD 1 = PAGE_SIZE_512 2 = PAGE_SIZE_1024 3 = PAGE_SIZE_2048 4 = PAGE_SIZE_4096 5 = PAGE_SIZE_8192 6 = PAGE_SIZE_RSVD2 7 = PAGE_SIZE_RSVD3
15:14	0x0	SKIP_SPARE_SEL: Size in granularity of 4 bytes to skipped for spare access 0 = SKIP_SPARE_SIZE_4 1 = SKIP_SPARE_SIZE_8 2 = SKIP_SPARE_SIZE_12 3 = SKIP_SPARE_SIZE_16
13	0x0	DEBUG_MODE: Debug mode selection for hardware debug
12:9	0x0	DEBUG_SEL: Debug selection for hardware debug
8:0	0x0	TAG_BYTE_SIZE: Block Size in bytes for TAG data from spare area of flash. This is used for specifying the size of the TAG Block data bytes to be move/from to spare area. Used when B_VALID is true. Specified in bytes (n-1 encoding)

29.4.6 NAND_TIMING_0

Control the flash interface signals for asynchronous timings

where n - value programmed in the respective field of timing register.

NAND Timing Register

Offset: 014h | Read/Write: R/W | Reset: 0b00000000000000000000000xx000000

Bit	Reset	Description
31:28	0x0	TRP_RESP_CNT: Read pulse width(RE Low time)timing for status read cycles Generated timing = (n+1) * NAND_CLK_PERIOD ns GUIDELINE: for tRP_RESP/tRP timing non-EDO mode: Max(tRP, tREA) timing + 6ns (round trip delay) EDO mode: tRP timing from flash datasheet Notes: (1)"round trip delay" to account for - REN out PAD delay + REN out board delay + DATA driven OUT from flash to chip input + DATA INPUT pad delay. (2)For EDO modes - since controller latches data without regard to `nRE' (REN) posedge tREA, round trip delay factors need not be considered.
27:24	0x0	TWB_CNT: WE High to RBSY low asserted (by flash) timing Generated timing = (n+1) * NAND_CLK_PERIOD ns. GUIDELINE: Refer to tWB timing from flash datasheet
23:20	0x0	TCR_TAR_TRR_CNT: RBSY High to RE low timing Generated timing = (n+3) * NAND_CLK_PERIOD ns. GUIDELINE: Program Max(tCR, tAR, tRR) timings from flash data sheet
19:16	0x0	TWHR_CNT: WE High to RE Low timing - Status Read Cycles Generated timing = (n+1) * NAND_CLK_PERIOD ns. GUIDELINE: Refer to tWHR timing from flash data sheet
15:14	0x0	TCS_CNT: CLE/ALE Setup/Hold time. Generated timing = (n+2) * NAND_CLK_PERIOD ns. GUIDELINE: Program Max(tCS, tCH, tALS, tALH) timings from flash datasheet This timing is met timing requirements. 1. from CE Low -> WE posedge of CLE/ALE. 2. from WE posedge of CLE to-> WE posedge of ALE. Please note that this amounts to, 1(above case): (n+2)*NAND_CLK_PERIOD + tWP(of following CLE/ALE cycle) 2(above case): tWH(hold time of current CLE cycle) + (n+2).NAND_CLK_PERIOD of CLE cycle (n+2)*NAND_CLK_PERIOD of ALE cycle + tWP(pusle width of ALE cycle).
13:12	0x0	TWH_CNT: Write pulse HOLD time Generated timing = (n+1) * NAND_CLK_PERIOD ns. GUIDELINE: Refer to tWH timing from flash datasheet

Bit	Reset	Description
11:8	0x0	TWP_CNT: Write pulse width time Generated timing = (n+1) * NAND_CLK_PERIOD ns. GUIDELINE: Refer to tWP timing from flash datasheet
5:4	0x0	TRH_CNT: Read pulse HOLD time Generated timing = (n+1) * NAND_CLK_PERIOD ns. GUIDELINE: Refer to tRH timing from flash datasheet
3:0	0x0	TRP_CNT: Read pulse width(RE Low time)timing for Data read cycles Generated timing = (n+1) * NAND_CLKS, where n - value programmed in the tRP_RESP_CNT field of timing register. GUIDELINE: tRP_RESP/tRP timing register programming non-EDO mode: Max(tRP, tREA) timing + 6ns (round trip delay) EDO mode: tRP timing Notes: (1) "round trip delay" to account for - REN out PAD delay + REN out board delay + DATA driven OUT from flash to chip input + DATA INPUT pad delay. (2) For EDO modes - since controller latches data without regard to `nRE' (REN) posedge tREA, round trip delay factors need not be considered.

29.4.7 NAND_RESP_0

Staging register meant for PIO mode of transfer Contents should be written by CPU access to this register for write transfer. Contents are readable by CPU access to this register after read transfer.

NAND Response Register

Offset: 018h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	BYTE3: Write/Response data byte 3 (MSB)
23:16	0x0	BYTE2: Write/Response data byte 2
15:8	0x0	BYTE1: Write/Response data byte 1
7:0	0x0	BYTE0: Write/Response data byte 0 (LSB)

29.4.8 NAND_TIMING2_0

Controls the flash interface signals asynchronous timings.

NAND Timing2 Register

Offset: 01ch | Read/Write: R/W | Reset: 0b0000

Bit	Reset	Description
3:0	0x0	TADL_CNT: WE posedge of address cycle to WE posedge of data cycle GUIDELINE: Refer to tADL timing from flash datasheet Please note that timing generated from controller is for the duration from ALE low to WP low. Generated timing = (TWH+TADL+TWP+6) * NAND_CLK_PERIOD ns

29.4.9 NAND_CMD_REG1_0

Command cycle generation use these during COMMAND1 time:

- CLE cycle1 command value setting

NAND Command1 Register

Offset: 020h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	CMD_BYTE3: Command byte 3(MSB)
23:16	0x0	CMD_BYTE2: Command byte 2
15:8	0x0	CMD_BYTE1: Command byte 1
7:0	0x0	CMD_BYTE0: Command byte 0(LSB)

29.4.10 NAND_CMD_REG2_0

Command cycle generation use these during COMMAND2 time:

- CLE cycle2 command value setting

NAND Command2 Register

Offset: 024h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	CMD_BYTE3: Command byte 3(MSB)
23:16	0x0	CMD_BYTE2: Command byte 2
15:8	0x0	CMD_BYTE1: Command byte 1
7:0	0x0	CMD_BYTE0: Command byte 0(LSB)

29.4.11 NAND_ADDR_REG1_0

Address cycle generation uses these bytes:

- ALE cycle value setting

NAND Address1 Register

Offset: 028h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	ADDR_BYTE3: Address byte 3
23:16	0x0	ADDR_BYTE2: Address byte 2
15:8	0x0	ADDR_BYTE1: Address byte 1
7:0	0x0	ADDR_BYTE0: Address byte 0 (LSB)

29.4.12 NAND_ADDR_REG2_0

Address cycle generation uses these bytes:

- ALE cycle value setting

NAND Addr2 Register

Offset: 02ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	ADDR_BYTE7: Address byte 3
23:16	0x0	ADDR_BYTE6: Address byte 2
15:8	0x0	ADDR_BYTE5: Address byte 1
7:0	0x0	ADDR_BYTE4: Address byte 0 (LSB)

29.4.13 NAND_DMA_MST_CTRL_0

NAND controller DMA interface is configured with this register. Number of page transfers DMA can be initiated only ONCE.

Maximum number of pages transferrable directly depends on the maximum DMA length of 64K bytes.

NAND DMA Master Control Register

Offset: 030h | Read/Write: R/W | Reset: 0b00100100xxx0xxxxxxxxxxxxxxxxxx00

Bit	Reset	Description
31	0x0	DMA_GO: NAND DMA interface enable for data transfers. Hardware clears when programmed length of data transfer is completed. 0 = DISABLE 1 = ENABLE
30	0x0	DIR: DMA data transfer direction. 0 = DMA_RD : 1 = DMA_WR : Read from system and write to flash

Bit	Reset	Description
29	0x1	DMA_PERF_EN: DMA performance feature enable. As soon as the Error vectors equal to BURST SIZE programmed received DMA suspends current data transfers and moves to Error vector transfer and waits till that page decode is completed. Potentially if Error vectors are received around each 512 sub-page boundary this could cause stall of next page READ data transfers causing performance degradation. To take advantage of PIPELINE_EN ECC decoder pipeline capability this should be enabled. 0 = DISABLE : 1 = ENABLE
28	0x0	IE_DMA_DONE: DMA transfer completion interrupt enable 0 = DISABLE 1 = ENABLE
27	0x0	REUSE_BUFFER: Reuse buffer for storing error vectors. 0 = DISABLE, increments the Error Vector destination address continuously till the total DMA transfer size is done 1 = ENABLE, same buffer is used for storing the error vectors for every page decoded
26:24	0x4	BURST_SIZE: DMA burst size selection 0 = BURST_RSVD1 1 = BURST_RSVD2 2 = BURST_1WORDS 3 = BURST_4WORDS 4 = BURST_8WORDS 5 = BURST_16WORDS 6 = BURST_RSVD3 7 = BURST_RSVD4
20	0x0	IS_DMA_DONE: 1 = DMA transfer completed interrupt status. This is set ONLY when not running in COMMAND QUEUE MODE
2	0x0	DMA_EN_A: Enable DMA transfer for Data (A) 0 = DISABLE 1 = ENABLE
1	0x0	DMA_EN_B: Enable DMA transfer for TAG/Spare (B) 0 = DISABLE 1 = ENABLE

29.4.14 NAND_DMA_CFG_A_0

DMA transfer length meant for main area in flash.

NAND DMA Configuration Register for main area

Offset: 034h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	DMA_BLOCK_SIZE_A: Normal mode: DMA Data Block size in Bytes(N-1) value Sgdma mode : no. of entries in the main descriptor table (N-1)

29.4.15 NAND_DMA_CFG_B_0

DMA transfer length meant for spare area in flash.

NAND DMA Configuration Register for spare area

Offset: 038h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	DMA_BLOCK_SIZE_B: DMA TAG Block size in Bytes(N-1) value

29.4.16 NAND_FIFO_CTRL_0

Indicates FIFO status useful for debug. Also each of these FIFO buffers can be flushed out to come out of abnormal conditions (e.g., controller hang).

NAND FIFO Control/Status Register

Offset: 03ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxx0000

Bit	R/W	Reset	Description
15	RO	X	LL_BUF_EMPTY: 1 = Indicates Command queue FIFO Empty
14	RO	X	LL_BUF_FULL: 1 = Indicates Command queue FIFO Full
13	RO	X	FIFO_A_EMPTY: 1 = Indicates Data FIFO Empty
12	RO	X	FIFO_A_FULL: 1 = Indicates Data FIFO Full
11	RO	X	FIFO_B_EMPTY: 1 = Indicates TAG FIFO Empty
10	RO	X	FIFO_B_FULL: 1 = Indicates TAG FIFO Full
9	RO	X	FIFO_C_EMPTY: 1 = Indicates ECC FIFO Empty
8	RO	X	FIFO_C_FULL: 1 = Indicates ECC FIFO Full
3	RW	0x0	LL_BUF_CLR: field set to "CLEAR_ALL_FIFO" flushes all the buffers(i.e.,LL_BUF,FIFO_A,FIFO_B,FIFO_C) 0 = CLEAR_NO_FIFO 1 = CLEAR_ALL_FIFO
2	RW	0x0	FIFO_A_CLR: Flush the DATA FIFO contents
1	RW	0x0	FIFO_B_CLR: Flush the TAG FIFO contents
0	RW	0x0	FIFO_C_CLR: Flush the ECC FIFO contents

29.4.17 NAND_DATA_BLOCK_PTR_0

NAND Data Block Address Register

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	DMA_DATA_BLOCK_PTR: DMA data block source/destination address pointer Buffer format in descriptor mode SGDMA_EN=1 Addr0(Byte aligned, 32-bit memory address | Length0(Byte granularity) | Addr1(Byte aligned, 32-bit memory address | Length1(Byte granularity) |.... | Buffer format in regular mode SGDMA_EN=0 Data word0(4bytes) Data word1(4bytes) Data word2(4bytes) ....

29.4.18 NAND_TAG_PTR_0

NAND TAG Block Address Register

Offset: 044h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	DMA_TAG_PTR: DMA TAG block source/destination address pointer Buffer format in descriptor mode SGDMA_EN=1 Addr0(Byte aligned, 32-bit memory address | Length0(Byte granularity) | Addr1(Byte aligned, 32-bit memory address | Length1(Byte granularity) |.... | Buffer format in regular mode SGDMA_EN=0 Tag word0(4bytes) Tag word1(4bytes) Tag word2(4bytes) ....

29.4.19 NAND_ECC_PTR_0

NAND ECC Block Address Register

Offset: 048h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	DMA_ECC_PTR: DMA Error vector destination address pointer

29.4.20 NAND_DEC_STATUS_0

NAND Decode Status Register

Offset: 04ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	ERR_PAGE_NUMBER: Indicates the reference to the PAGE for Error Correction to be applied. Valid when IS_ECC_ERROR is generated
23:16	X	ERR_COUNT: No. of Errors occurred in main block READ data plus TAG read data when corresponding features are enabled.

Bit	Reset	Description
15:8	X	SUB_PAGE_FAIL: Indicates sub-page decode failure within a page size. When decode failure is observed software can use to figure out which sub-page (512 byte) decode failure. for ex: of 2K page size selection, bit 0 - first sub-page bit 1 - second sub-page bit 2 - third sub-page bit 3 - fourth sub-page and so on as applicable
1	X	A_ECC_FAIL: 0 = Main block data decode without decode fail
0	X	B_ECC_FAIL: 0 = Tag block data decode without decode fail

29.4.21 NAND_HWSTATUS_CMD_0

NAND Hardware Status Command Register

Offset: 050h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	HWSTATUS_CMD: Command byte value used for READ STATUS commands when automatic hardware RBSY_CHK or RD_STATUS_CHK are enabled.

29.4.22 NAND_HWSTATUS_MASK_0

NAND Read Status Mask Register

Offset: 054h | Read/Write: R/W | Reset: 0b11111111111000000100000001000000

Bit	Reset	Description
31:24	0xff	RDSTATUS_MASK: 8 bit Mask value to extract the correct bit fields from READ STATUS information for RD_STATUS_CHK
23:16	0xe0	RDSTATUS_EXP_VAL: 8 bit expected RD STATUS VALUE for RD_STATUS_CHK
15:8	0x40	RBSY_MASK: 8 bit Mask value to extract the correct bit fields from READ STATUS information for RBSY_CHK
7:0	0x40	RBSY_EXP_VAL: 8 bit expected RD STATUS VALUE for RBSY_CHK

29.4.23 NAND_LL_CONFIG_0

NAND Command Queue Configuration Register

Offset: 058h | Read/Write: R/W | Reset: 0b0xxxxxxxxxxx1100xxxx000000000000

Bit	Reset	Description
31	0x0	LL_START: Command Queue feature enable. Hardware clears when command queue data and flash operations are completed. 0 = DISABLE 1 = ENABLE
19	0x1	WORD_CNT_STATUS_EN: Enable word count status update in LL_STATUS register 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
18:16	0x4	BURST_SIZE: DMA burst size selection for Command Queue data requests 0 = BURST_RSVD1 1 = BURST_RSVD2 2 = BURST_1WORDS 3 = BURST_4WORDS 4 = BURST_8WORDS 5 = BURST_16WORDS 6 = BURST_RSVD3 7 = BURST_RSVD4
11:0	0x0	LL_LENGTH: Command queue up word length programmed is parsed and `START is done when the execution is complete. However, when errors are detected for any case of the flash operation failure command queue execution is aborted immediately before this length.

29.4.24 NAND_LL_PTR_0

NAND Command Queue Address Register

Offset: 05ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:2	0x0	LL_PTR: Command queue data pointer Register

29.4.25 NAND_LL_STATUS_0

NAND Command Queue Status Register

Offset: 060h | Read/Write: RO | Reset: 0bxxxxxxxx00xxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	LL_PKT_ID: Command queue PACKET ID completed at this time. Software has write access to this bit field position so that any time software can clear this field. Also NAND controller reset will reset this status.
23	0x0	IS_LL_DONE: Interrupt status of LL_DONE {Read Only}
22	X	IS_LL_ERR: Interrupt status of LL_ERR {Read Only}
19:16	X	LL_LENGTH_LAST_PKT: Command queue Word length of last packet executed in the queue. Please note that WORD_CNT_STATUS_EN in LL_CONFIG should be enabled for this status update
11:0	X	LL_LENGTH_DONE: Command queue Word length(32-bit) completed till this time. Please note that WORD_CNT_STATUS_EN in LL_CONFIG should be enabled for this status update

29.4.26 NAND_LOCK_CONTROL_0

Flash erase operation for selected [7:0] apertures are discarded.

NAND Memory Lock Control Register

Offset: 064h | Read/Write: R/W | Reset: 0b000000000

Bit	Reset	Description
8	0x0	IE_LOCK_ERR: Interrupt enable on memory range match. 0 = DISABLE 1 = ENABLE
7	0x0	LOCK_APER_EN7: Enable lock feature for selected apertures 7 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START7, LOCK_APER_END7, LOCK_APER_CHIPID7 can't be programmed once this SET 0 = DISABLE 1 = ENABLE
6	0x0	LOCK_APER_EN6: Enable lock feature for selected apertures 6 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START6, LOCK_APER_END6, LOCK_APER_CHIPID6 can't be programmed once this SET 0 = DISABLE 1 = ENABLE
5	0x0	LOCK_APER_EN5: Enable lock feature for selected apertures 5 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START5, LOCK_APER_END5, LOCK_APER_CHIPID5 can't be programmed once this SET 0 = DISABLE 1 = ENABLE
4	0x0	LOCK_APER_EN4: Enable lock feature for selected apertures 4 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START4, LOCK_APER_END4, LOCK_APER_CHIPID4 can't be programmed once this SET 0 = DISABLE 1 = ENABLE
3	0x0	LOCK_APER_EN3: Enable lock feature for selected apertures 3 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START3, LOCK_APER_END3, LOCK_APER_CHIPID3 can't be programmed once this SET 0 = DISABLE 1 = ENABLE
2	0x0	LOCK_APER_EN2: Enable lock feature for selected apertures 2 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START2, LOCK_APER_END2, LOCK_APER_CHIPID2 can't be programmed once this SET 0 = DISABLE 1 = ENABLE
1	0x0	LOCK_APER_EN1: Enable lock feature for selected apertures 1 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START1, LOCK_APER_END1, LOCK_APER_CHIPID1 can't be programmed once this SET 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	0x0	LOCK_APER_EN0: Enable lock feature for selected apertures 0 Can be set only once register field, hardware reset OR controller reset ONLY can disable this feature. LOCK_APER_START0, LOCK_APER_END0, LOCK_APER_CHIPID0 can't be programmed once this SET 0 = DISABLE 1 = ENABLE

29.4.27 NAND_LOCK_STATUS_0

Status indicating which aperture matched.

Offset: 068h | Read/Write: R/W | Reset: 0b000000000

Bit	R/W	Reset	Description
8	RW	0x0	IS_LOCK_ERR: 1 = Memory protection error detected check LOCK_STATUS register to identify which aperture matched.
7	RO	0x0	LOCK_APER_STATUS7: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
6	RO	0x0	LOCK_APER_STATUS6: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
5	RO	0x0	LOCK_APER_STATUS5: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
4	RO	0x0	LOCK_APER_STATUS4: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
3	RO	0x0	LOCK_APER_STATUS3: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
2	RO	0x0	LOCK_APER_STATUS2: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
1	RO	0x0	LOCK_APER_STATUS1: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information
0	RO	0x0	LOCK_APER_STATUS0: Respective bit is set by hardware on protection range detection. Write 1 to clear IS.LOCK_ERR will clear this status information

29.4.28 NAND_LOCK_APER_START0_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0 Start Address Register

Offset: 06ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.29 NAND_LOCK_APER_START1_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture1 Start Address Register

Offset: 070h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.30 NAND_LOCK_APER_START2_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture2 Start Address Register

Offset: 074h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.31 NAND_LOCK_APER_START3_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture3 Start Address Register

Offset: 078h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.32 NAND_LOCK_APER_START4_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture4 Start Address Register

Offset: 07ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.33 NAND_LOCK_APER_START5_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture5 Start Address Register

Offset: 080h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.34 NAND_LOCK_APER_START6_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture6 Start Address Register

Offset: 084h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.35 NAND_LOCK_APER_START7_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture7 Start Address Register

Offset: 088h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.36 NAND_LOCK_APER_END0_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

Offset: 08ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.37 NAND_LOCK_APER_END1_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

Offset: 090h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.38 NAND_LOCK_APER_END2_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

Offset: 094h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.39 NAND_LOCK_APER_END3_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

Offset: 098h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.40 NAND_LOCK_APER_END4_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

Offset: 09ch | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.41 NAND_LOCK_APER_END5_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

Offset: 0a0h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.42 NAND_LOCK_APER_END6_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

Offset: 0a4h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.43 NAND_LOCK_APER_END7_0

Program this in the format of "Row Address" NAND flash memory for protection.

NAND Memory Lock Aperture0-7 End Address Register

Offset: 0a8h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	ADDR

29.4.44 NAND_LOCK_APER_CHIPID0_0

NAND Memory Lock Aperture0-7 Chip Select Register

Offset: 0ach | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE

Bit	Reset	Description
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

29.4.45 NAND_LOCK_APER_CHIPID1_0

NAND Memory Lock Aperture0-7 Chip Select Register

Offset: 0b0h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

29.4.46 NAND_LOCK_APER_CHIPID2_0

NAND Memory Lock Aperture0-7 Chip Select Register

Offset: 0b4h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

29.4.47 NAND_LOCK_APER_CHIPID3_0

NAND Memory Lock Aperture0-7 Chip Select Register

Offset: 0b8h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE

Bit	Reset	Description
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

29.4.48 NAND_LOCK_APER_CHIPID4_0

NAND Memory Lock Aperture0-7 Chip Select Register

Offset: 0bch | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

29.4.49 NAND_LOCK_APER_CHIPID5_0

NAND Memory Lock Aperture0-7 Chip Select Register

Offset: 0c0h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

29.4.50 NAND_LOCK_APER_CHIPID6_0

NAND Memory Lock Aperture0-7 Chip Select Register

Offset: 0c4h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE

Bit	Reset	Description
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

29.4.51 NAND_LOCK_APER_CHIPID7_0

NAND Memory Lock Aperture0-7 Chip Select Register

Offset: 0c8h | Read/Write: R/W | Reset: 0bxxxxxxxx

Bit	Reset	Description
7	X	CS7: Memory Protection of aperture[0-7] valid for Chip select7 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select7 1 = ENABLE
6	X	CS6: Memory Protection of aperture[0-7] valid for Chip select6 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select6 1 = ENABLE
5	X	CS5: Memory Protection of aperture[0-7] valid for Chip select5 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select5 1 = ENABLE
4	X	CS4: Memory Protection of aperture[0-7] valid for Chip select4 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select4 1 = ENABLE
3	X	CS3: Memory Protection of aperture[0-7] valid for Chip select3 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select3 1 = ENABLE
2	X	CS2: Memory Protection of aperture[0-7] valid for Chip select2 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select2 1 = ENABLE
1	X	CS1: Memory Protection of aperture[0-7] valid for Chip select1 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select1 1 = ENABLE
0	X	CS0: Memory Protection of aperture[0-7] valid for Chip select0 0 = DISABLE : Memory Protection of aperture[0-7] not valid for Chip select0 1 = ENABLE

29.4.52 NAND_BCH_CONFIG_0

NAND BCH Error Correction Control Register

Offset: 0cch | Read/Write: R/W | Reset: 0b11111xxx11111x000x000xxx0

Bit	Reset	Description
24:20	0x1f	ERR_ATTN_LVL_TAG: Trigger for correctable error Interrupts. Mechanism for SW to get an idea on error pattern development over a period of usage. NAND controller will trigger interrupt if the current main page read transfer resulted in correctable errors reached this trigger value. 0 = CORR_ERRS_0 1 = CORR_ERRS_1 2 = CORR_ERRS_2 3 = CORR_ERRS_3 4 = CORR_ERRS_4 5 = CORR_ERRS_5 6 = CORR_ERRS_6 7 = CORR_ERRS_7 8 = CORR_ERRS_8 9 = CORR_ERRS_9 10 = CORR_ERRS_10 11 = CORR_ERRS_11 12 = CORR_ERRS_12 13 = CORR_ERRS_13 14 = CORR_ERRS_14 15 = CORR_ERRS_15 16 = CORR_ERRS_16 17 = CORR_ERRS_17 18 = CORR_ERRS_18 19 = CORR_ERRS_19 20 = CORR_ERRS_20 21 = CORR_ERRS_21 22 = CORR_ERRS_22 23 = CORR_ERRS_23 24 = CORR_ERRS_24
16:12	0x1f	ERR_ATTN_LVL_DATA: Trigger for correctable error Interrupts. Mechanism for SW to get an idea on error pattern development over a period of usage. NAND controller will trigger interrupt if the current main page read transfer resulted in correctable errors reached this trigger value. 0 = CORR_ERRS_0 1 = CORR_ERRS_1 2 = CORR_ERRS_2 3 = CORR_ERRS_3 4 = CORR_ERRS_4 5 = CORR_ERRS_5 6 = CORR_ERRS_6 7 = CORR_ERRS_7 8 = CORR_ERRS_8 9 = CORR_ERRS_9 10 = CORR_ERRS_10 11 = CORR_ERRS_11 12 = CORR_ERRS_12 13 = CORR_ERRS_13 14 = CORR_ERRS_14 15 = CORR_ERRS_15 16 = CORR_ERRS_16 17 = CORR_ERRS_17 18 = CORR_ERRS_18 19 = CORR_ERRS_19 20 = CORR_ERRS_20 21 = CORR_ERRS_21 22 = CORR_ERRS_22 23 = CORR_ERRS_23 24 = CORR_ERRS_24

Bit	Reset	Description
10:8	0x0	BCH_TAG_TVALUE: BCH error correction strength selection for Tag 24 single bit random errors per codeword 0 = BCH_TVAL4 : 4 single bit random errors per codeword 1 = BCH_TVAL8 : 8 single bit random errors per codeword 2 = BCH_TVAL14 : 14 single bit random errors per codeword 3 = BCH_TVAL16 : 16 single bit random errors per codeword 4 = BCH_TVAL24
6:4	0x0	BCH_TVALUE: BCH error correction strength selection 24 single bit random errors per codeword 0 = BCH_TVAL4 : 4 single bit random errors per codeword 1 = BCH_TVAL8 : 8 single bit random errors per codeword 2 = BCH_TVAL14 : 14 single bit random errors per codeword 3 = BCH_TVAL16 : 16 single bit random errors per codeword 4 = BCH_TVAL24
0	0x0	BCH_ECC: BCH encoder & decoder is enabled 0 = DISABLE : BCH encoder & decoder is not enabled 1 = ENABLE

29.4.53 NAND_BCH_DEC_RESULT_0

NAND BCH/RS/Hamming extended decode status information

- CORRFAIL_ERR flag is added to NAND_ISR register to indicate that out of total DMA transfers of specified pages there are correctable OR un-correctable errors. Please check the description field below. If there are no correctable/un-correctable errors this bit will not be SET.
- New register NAND_DEC_RESULT indicates the count of pages resulted in correctable/un-correctable errors.
- New register NAND_DEC_STAT_BUF. Reading this register will fetch out an entry from decode status buffer. Each entry of this buffer relate to page which resulted in either correctable/un-correctable error. Please check the description of this entry described as NAND_DEC_STAT_EXT below. It is up to software how this can be used; hopefully this serves for all purposes.

Example

If there are 4 pages resulted in correctable errors and 2 pages of uncorrectable errors out of total DMA transfer of 64 pages, PAGE_COUNT in NAND_DEC_RESULT will indicate as PAGE_COUNT=6. Software should read these 6 entries reading NAND_DEC_STAT_BUF register by CPU access and process it for further action.

Offset: 0d0h | Read/Write: RO | Reset: 0bxxxxxxxxx

Bit	Reset	Description
8	X	CORRFAIL_ERR: 1 = Correctable OR Un-correctable errors occurred in the DMA transfer without regard to HW_ERR_CORRECTION feature is enabled or not. Use extended decode results in NAND_DEC_RESULT and NAND_DEC_STATUS_EXT to figure out further action for block replacement/wear leveling during file system management for software.
7:0	X	PAGE_COUNT: No. of pages resulted either in un-correctable or correctable errors

29.4.54 NAND_BCH_DEC_STATUS_BUF_0

For page sizes upto 4k one entry per page exists in the Decode status buffer

For page size 8k, two entries exists in Decode status buffer.

Upon DMA complete software can read this buffer mapped to this register address to gather error statistics.

ECC Decode status Register

Offset: 0d4h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	FAIL_SEC_FLAG: Sector wise un-correctable error indicator Bit 31 = 1, sector 7 has un-correctable errors Bit 31 = 0, sector 7 has no un-correctable errors ... Bit 24 = 1, sector 0 has un-correctable errors Bit 24 = 0, sector 0 has no un-correctable errors
23:16	X	CORR_SEC_FLAG: Sector wise correctable error indicator Bit 23 = 1, sector 7 has correctable errors Bit 23 = 0, sector 7 has no correctable errors ... Bit 16 = 1, sector 0 has correctable errors Bit 16 = 0, sector 0 has no correctable errors
14	X	FAIL_TAG: Spare area error decode resulted in un-correctable errors in case of RS/Hamming ECC selection. For BCH this field is not applicable.
13	X	CORR_TAG: Spare area error decode resulted in correctable errors in case of RS/Hamming ECC selection. For BCH this field is not applicable.
12:8	X	MAX_CORR_CNT: Maximum no. of correctable errors occurred out of all sectors. For example of 2K page, if sector 0 has 2 correctable errors and sector3 has 4 errors MAX_ERR_CNT will reflect as 4
7:0	X	PAGE_NUMBER: Page number which resulted in either correctable/un-correctable errors 0 to 63 indication for 64 pages of DMA transfer.

29.4.55 NAND_CONFIG2_0

Field interpretation of 2nd word in 8k page

New Control register for features added in Tegra 3

Offset: 0d8h l Read/Write: R/W l Reset: 0b00xxxxxxxxxxxx0000000000000x000

Bit	Reset	Description
31	0x0	ABORT: 0 = HW clears when abort is completed and controller has gone back to idle and ready for next transfer. 1 = Written by software, set triggers abort.
30	0x0	FORCE_ABORT: intended to be used only in specific cases. Typically software would like this set to 0.
16	0x0	CODEWORD_SEL: Codeword selection for ECC computation as well as minimum unit for partial page read. No of sectors read during partial page read is determined by this selection. 14 bit polynomial, minimum ECC unit is 1k bytes 0 = BCH_512 : 13 bit polynomial, minimum ECC unit is 512 bytes This is default selection i.e. backward compatible to Tegra 2. 1 = BCH_1024
15:12	0x0	SECTOR_START: Reserved
11:8	0x0	SECTOR_CNT: Reserved
7	0x0	TIMING_SEL: Implement timing as per proposed new timing registers. 0 = DISABLE : Implement timings as per NAND_TIMING, NAND_TIMING2 registers 1 = ENABLE

Bit	Reset	Description
6	0x0	CLK_OFF: power saving enable by turning off clock in DDR interface Clock will be turned during unused time of Write operation Feature applicable for SYNC_DDR mode of Interface selection 0 = DISABLE : Clock will be continuous during the entire period of write operation 1 = ENABLE
5	0x0	SYNC_DDR: DDR mode of operation, as defined by ONFI standard DQS toggles but there is clock in interface. During Command/Address phase SDR mode of transfer is done. I.e One Address/Command byte per clock During Data transfer phase data is transferred on every edge of clock/DQS 0 = DISABLE : 1 = ENABLE
4	0x0	TOGGLE_DDR: Synchronous DDR mode of operation, as defined by Samsung. During Command/Address phase ASYNC mode of transfer is done. During Data transfer phase data is transferred on every edge of DQS 0 = DISABLE : 1 = ENABLE
2	0x0	PARTIAL_PAGE: Reserved 0 = DISABLE 1 = ENABLE
1	0x0	SGDMA_EN: Scatter Gather DMA feature is enabled. NAND_DATA_BLOCK_PTR, NAND_TAG_PTR indicates buffer address of respective descriptor buffers. Descriptor buffer consists of, Addr0(Byte aligned, 32-bit memory address | Length0(Byte granularity) | Addr1(Byte aligned, 32-bit memory address | Length1(Byte granularity) |.... | A single descriptor buffer fetch should result in completing the total DMA length as programmed in DMA_CNFGA DMA_CNFGB registers. 0 = DISABLE : Scatter Gather DMA feature is turned off. NAND_DATA_BLOCK_PTR, NAND_TAG_PTR indicates source/destination buffer address which are word aligned 1 = ENABLE
0	0x0	PROG_OFFSET_EN: Enable software control for Tegra 3. Features position bytes in spare region (1) Offset locations programmed in MAIN_ECC_OFFSET, TAG_OFFSET, TAG_ECC_OFFSET will be used to position the bytes in spare region for read/write/ (2) Tag ECC is done separately. Tag bytes, and parity bytes will be positioned in spare as programmed in OFFSET registers Without BCH_ECC, tag only transfers are also possible. ECC strength of Tag is determined by the BCH_TAG_TVALUE setting Tag transfer size is determined by the TAG_BYTE_SIZE and B_valid bit fields in NAND_CONFIG, NAND_COMMAND registers respectively. Main only transfers are possible clearing B_VALID bit field just like in Tegra 2 devices. 0 = DISABLE : hardware controller behavior as in Tegra 2 devices. (1) I.e. Based on the skip spare selection following order is followed. Skip bytes -> Tag Bytes -> ECC bytes of Main and Tag together (2) Tag ECC computation is done along with Main last unit of data 1 = ENABLE

29.4.56 NAND_MAIN_ECC_OFFSET_0

Programmable offset for Main data ecc bytes in spare. Offset location is w.r.t start of column address of spare region.

Example: from 4th byte location in spare required settings of START_ADDR = 0x4

Offset: 0dch l Read/Write: R/W l Reset: 0b000000000000

Bit	Reset	Description
11:0	0x0	START_ADDR: Total size of ECC bytes granularity is in terms byte for 8-bit device and 2-byte for 16 bit device. Since this is dictated by flash access width

29.4.57 NAND_TAG_OFFSET_0

Programmable offset for Tag bytes in spare. Offset location is w.r.t start of column address of spare region.

Offset: 0e0h l Read/Write: R/W l Reset: 0b000000000000

Bit	Reset	Description
11:0	0x0	START_ADDR: Total size of Tag bytes granularity is in terms byte for 8-bit device and 2-byte for 16 bit device. Since this is dictated by flash access width |

29.4.58 NAND_TAG_ECC_OFFSET_0

Programmable offset for Tag ecc bytes in spare. Offset location is w.r.t start of column address of spare region.

Offset: 0e4h l Read/Write: R/W l Reset: 0b000000000000

Bit	Reset	Description
11:0	0x0	START_ADDR: Total size of Tag ecc bytes granularity is in terms byte for 8-bit device and 2-byte for 16 bit device. Since this is dictated by flash access width

29.4.59 NAND_RESP2_0

Staging register meant for PIO mode of transfer. Contents should be written by CPU access to this register for write transfer. Contents are readable by CPU access to this register after read transfer.

NAND Response2 Register

Offset: 0e8h l Read/Write: R/W l Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	BYTE3: Write/Response data byte 3 (MSB)
23:16	0x0	BYTE2: Write/Response data byte 2
15:8	0x0	BYTE1: Write/Response data byte 1
7:0	0x0	BYTE0: Write/Response data byte 0 (LSB)

29.4.60 NAND_RANDOM_RD_0

Defining register for Random Data input/output command sequences required to seek to respective column address location for sector wise read/write operation.

Register for Random Data Output command values

Offset: 0ech l Read/Write: R/W l Reset: 0b0000010111100000

Bit	Reset	Description
15:8	0x5	CMD1_CODE
7:0	0xe0	CMD2_CODE

29.4.61 NAND_RANDOM_WR_0

Register for Random Data Input command value

Offset: 0f0h l Read/Write: R/W l Reset: 0b10000101

Bit	Reset	Description
7:0	0x85	CMD1_CODE

29.4.62 NAND_ASYNC_TIMING_0_0

Required for Asynchronous or all DDR modes.

ASYNC Timing register0

Offset: 0f4h l Read/Write: R/W l Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
27:24	0x0	TAR: Address cycle ALE low to data output time Covered by tRR timing during Data read operation Generated timing during Read ID, Max(tAR, tWHR) = (TAR_CNT+1) * NAND_CLK_PERIOD ns This timing parameter is not relavant for ONFI DDR mode, where in recommended programming value, TAR_CNT=0.
23:20	0x0	TCR: RSVD: Command cycle CE/CLE low to data output time Recommended programming value, TAR_CNT=0. Below equations indicate implemented timings for tCR/tCLR timings, which are not dependent on TCR value programmed. If tRR, tWHR, tWHR2 timings are met tCR/tCLR timings will be met Automatically. Generated timing, tCR = (TRR_CNT + 4) * NAND_CLK_PERIOD ns Generated timing during status read, tCLR = (TWHR_CNT + TRR_CNT +3) * NAND_CLK_PERIOD ns Generated timing during random read, tCLR = (TWHR2_CNT + 4) * NAND_CLK_PERIOD ns
19:16	0x0	TRR: Flash Ready to data output time Program this to meet timing requirement much as Max(tAR,tRR) from datasheet. Generated timing for all modes, tRR = (TRR_CNT + 4) * NAND_CLK_PERIOD ns
15:8	0x0	TWHR: WE High to data output time Generated timing for async and Toggle DDR modes, tWHR = (TWH_CNT + TCLH_CNT + TWHR_CNT + TRR_CNT + 5) * NAND_CLK_PERIOD ns. Generated timing for ONFI DDR modes, tWHR = (TCAD_CNT + TWHR_CNT + 4) * NAND_CLK_PERIOD ns.
7:0	0x0	TWB: WE High to RBSY low time Generated timing for all async and toggle DDR modes, tWB = (TWH_CNT + TCLH_CNT + TWB_CNT - 6) * NAND_CLK_PERIOD ns. Generated timing for ONFI DDR, tWB = (TCAD_CNT + TWB_CNT - 6) * NAND_CLK_PERIOD ns.

29.4.63 NAND_ASYNC_TIMING_1_0

Required for Asynchronous or all DDR modes.

ASYNC Timing register1

Offset: 0f8h l Read/Write: R/W l Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	TRHW: Data output to command, address, or data input time ONFI device specific. Generated timing for all async and toggle DDR modes, tRHW = (TRHW_CNT + TREH_CNT + TCLS_CNT + 4) * NAND_CLK_PERIOD ns. For ONFI DDR modes, tRHW = (TRHW_CNT + 1) * NAND_CLK_PERIOD ns.

Bit	Reset	Description
23:16	0x0	TWHR2: WE High to data output time for Random data out Relevant for Samsung new devices only, Generated timing, tWHR2 = (TCLH_CNT + TWH_CNT + TWHR2_CNT + 6) * NAND_CLK_PERIOD ns For ONFI DDR, recommended programming value TWHR2_CNT = 0.
15:8	0x0	TCS: Chip select setup time Generated timing for all ASYNC and toggle DDR, tCS = (TCS_CNT + TCLS_CNT + TWP_CNT + 4) * NAND_CLK_PERIOD ns Generated timing for ONFI DDR, tCS = (TCS_CNT + 1) * NAND_CLK_PERIOD ns If Flash is not ready busy and controller waiting for RBSY_CHK condition, that time will add up.
7:0	0x0	TADL: Address to data loading time Generated timing for ASYNC and toggle DDR, tADL = (TWH_CNT + TALH_CNT + TADL_CNT + 4) * NAND_CLK_PERIOD ns Generated timing for ONFI DDR, tADL = (TCAD_CNT + TADL_CNT + 4) * NAND_CLK_PERIOD ns

29.4.64 NAND_ASYNC_TIMING_2_0

Required for Asynchronous or all DDR modes.

ASYNC Timing register2

Offset: 0fch l Read/Write: R/W l Reset: 0b000000000000xxxx0000000000000000

Bit	Reset	Description
31:24	0x0	TCWAW: Command Write cycle to Address cycle time for Random data input Relevant for Samsung new devices only, Generated timing for async and toggle DDR modes, tCWAW = (TCWAW_CNT + TWH_CNT + TALS_CNT + TWP_CNT + 5) * NAND_CLK_PERIOD ns For ONFI DDR, recommended programming value TCWAW_CNT = 0.
23:20	0x0	TCH: RSVD Hardware implementation guarantees timing requirement. Generated timing for all ASYNC and toggle DDR, tCH = (TCLH_CNT + TWH_CNT + 3) * NAND_CLK_PERIOD ns Fixed generated timing for ONFI DDR, tCH = NAND_CLK_PERIOD/2 ns.
15:12	0x0	TCLH: CLE/CH hold time Generated timing for all ASYNC and toggle DDR, tCLH = (TCLH_CNT + TWH_CNT + 3) * NAND_CLK_PERIOD ns Fixed generated timing for ONFI DDR, tCLH = NAND_CLK_PERIOD/2 ns. Hence required programming value TCLH_CNT = 0
11:8	0x0	TCLS: CLE setup time Generated timing for all ASYNC and toggle DDR, tCLS = (TCLS_CNT + TWP_CNT + 2) * NAND_CLK_PERIOD ns Fixed generated timing for ONFI DDR, tCLS = NAND_CLK_PERIOD/2 ns. Hence required programming value TCLS_CNT = 0
7:4	0x0	TALH: ALE hold time Generated timing for all ASYNC modes, tALH = (TALH + TWH_CNT + 3) * NAND_CLK_PERIOD ns Generated timing for toggle DDR, tALH = (TCDQSS + TWH_CNT + 3) * NAND_CLK_PERIOD ns Fixed generated timing for ONFI DDR, tALH = NAND_CLK_PERIOD/2 ns. Hence required programming value TALH_CNT = 0
3:0	0x0	TALS: ALE setup time Generated timing for all ASYNC modes, tALS = (TALS_CNT + TWP_CNT + 2) * NAND_CLK_PERIOD ns Fixed generated timing for ONFI DDR, tALS = NAND_CLK_PERIOD/2 ns. Hence required programming value TALS_CNT = 0

29.4.65 NAND_ASYNC_TIMING_3_0

Required for Asynchronous or Toggle DDR modes.

ASYNC Timing register3

Offset: 100h l Read/Write: R/W l Reset: 0b00000000000000000000

Bit	Reset	Description
19:16	0x0	TREA: RE access time. RE low to valid dataout from device For Status read cycles, instead of TRP timing, TREA timing will be used. Generated timing for non-EDO mode, Max(tRP, tREA) = (TREA_CNT +1) * NAND_CLK_PERIOD ns Generated timing for EDO mode, tREA = (TREA_CNT +1) * NAND_CLK_PERIOD ns
15:12	0x0	TWH: Write cycle Hold time Generated timing, tWH = (TWH_CNT +1) * NAND_CLK_PERIOD ns.
11:8	0x0	TWP: Write cycle Pulse(low) time Generated timing, tWP = (TWP_CNT +1) * NAND_CLK_PERIOD ns.
7:4	0x0	TREH: Read cycle Hold time Generated timing, tREH = (TREH_CNT +1) * NAND_CLK_PERIOD ns. Fixed generated timing in toggle DDR mode, tREH = (NAND_CLK_PERIOD/2)ns
3:0	0x0	TRP: Read cycle Pulse(low) time Generated timing for non-EDO mode, Max(tRP, tREA) = (TRP_CNT +1) * NAND_CLK_PERIOD ns Generated timing for EDO mode, tRP = (TRP_CNT +1) * NAND_CLK_PERIOD ns. Fixed generated timing in toggle DDR mode, tRP = (NAND_CLK_PERIOD/2)ns

29.4.66 NAND_TDDR_TIMING_0_0

Required for Toggle DDR mode. Note that all ASYNC timing registers (ASYNC_TIMING_0-3) to be programmed along with DDR specific timing registers.

Toggle DDR Timing register0

Offset: 104h l Read/Write: R/W l Reset: 0b000000000000

Bit	Reset	Description
11:8	0x0	TRPSTH: Data Read Postamble Hold time Generated timing, tRPSTH = (TRPSTH_CNT +1) * NAND_CLK_PERIOD ns.
7:4	0x0	TRPST: Data Read Postamble time Generated timing, tRPST = (TRPST_CNT +2) * NAND_CLK_PERIOD ns. Right now round trip delay extends this a bit more.
3:0	0x0	TRPRE: Data Read Preamble time Generated timing, tRPRE = (TRPRE_CNT +1) * NAND_CLK_PERIOD ns.

29.4.67 NAND_TDDR_TIMING_1_0

Required for Toggle DDR mode. Note that all ASYNC timing registers (ASYNC_TIMING_0-3) to be programmed along with DDR specific timing registers.

Toggle DDR Timing register1

Offset: 108h l Read/Write: R/W l Reset: 0b0000000000000000xxxx000000000000

Bit	Reset	Description
31:24	0x0	TCDQSH: DQS hold time for data input finish Generated timing tCDQSH = (TWH_CNT + TCLH_CNT + TCDQSH_CNT +5) * NAND_CLK_PERIOD ns.
23:16	0x0	TCDQSS: DQS setup time for data input start Generated timing tCDQSS = TCDQSS_CNT * NAND_CLK_PERIOD ns.
11:8	0x0	TWPSTH: Data write Postamble Hold time Generated timing, tWPSTH = (TWPSTH_CNT + 1) * NAND_CLK_PERIOD ns.
7:4	0x0	TWPST: Data write Postamble time Generated timing, tWPST = (TWPST_CNT+ 4) * NAND_CLK_PERIOD ns.
3:0	0x0	TWPRE: Data Write Preamble time Generated timing tWPRE = (TCDQSS_CNT + TADL_CNT + TWPRE_CNT + 2) * NAND_CLK_PERIOD ns.

29.4.68 NAND_SDDR_TIMING_0_0

Required for SYNC_DDR mode. Note that ASYNC_TIMING0, ASYNC_TIMING1, ASYNC_TIMING2 registers to be programmed for sync DDR interface selection.

Synchronous DDR (standard DDR, ONFI DDR with clock) Timing register0

Offset: 10ch l Read/Write: R/W l Reset: 0b000000000000xxxx000000000000

Bit	Reset	Description
27:24	0x0	TCKWR: Data output end to W/R High time Generated timing, tCKWR = (TCKWR_CNT +1) * NAND_CLK_PERIOD ns.
23:16	0x0	TCAD: Command, Address, Data delay Command -> Command Address -> Address Command -> Address Address -> Command Command/Address -> start of data Generated timing tCAD = (TCAD_CNT+1) * NAND_CLK_PERIOD ns.
11:8	0x0	TWRCK: W/R low to data output cycle time Generated timing tWRCK = (TWRCK_CNT +1) * NAND_CLK_PERIOD ns.
7:4	0x0	TWPST: DQS write Postamble time Generated timing tWPST = 3 * NAND_CLK Required programmed values for all clock frequencies or devices TWPST_CNT = 0.
3:0	0x0	TWPRE: DQS Write Preamble time controls the timing parameters, tWRPRE, tDQSS Generated timing tDQSS = 1 * NAND_CLK. Generated timing tWRPRE = 3 * NAND_CLK. Required programmed values for all clock frequencies or devices TWPRE_CNT = 0.

29.4.69 NAND_SDDR_TIMING_1_0

Required for SYNC_DDR mode. Note that ASYNC_TIMING0, ASYNC_TIMING1, ASYNC_TIMING2 registers to be programmed for sync DDR interface selection.

Synchronous DDR (standard DDR, ONFI DDR with clock) Timing register1

Offset: 110h l Read/Write: R/W l Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	TCCS: Change column setup time Address cycle to data cycle for Random data commands Generated timing tCCS = (TCCS_CNT + TCAD_CNT + 4) * NAND_CLK_PERIOD ns.

29.4.70 NAND_FBIO_DQSIB_DLY_0

The FBIO_DQSIB_DELAY register sets the trimmers for inbound dqs delay on each of the inbound byte lanes.

FBIO configuration register

Offset: 114h l Read/Write: R/W l Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	CFG_DQSIB_DLY_BYTE_0: 47 = MAX

29.4.71 NAND_FBIO_QUSE_DLY_0

QUSE_LATE and QUSE_DLY determine how much added delay fbio should add to the QUSE path. QUSE needs to align (approximately) with the incoming DQS (DDR1) 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). For DDR1 only, QUSE_LATE provides finer granularity of 1/2 an m4clk cycle (1/2 bit time). The amount of delay we add will be 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.

QUSE_DLY controls trimmers in each byte lane. This also allows for variation in the propagation delay from the "common" pad macro where the QUSE logic lives out to each of the 4 byte lanes.

QUSE delay register

Offset: 118h l Read/Write: R/W l Reset: 0b00110100

Bit	Reset	Description
7:0	0x34	CFG_QUSE_DLY_BYTE_0: 47 = MAX

29.4.72 NAND_FBIO_CFG_0

FBIO configuration register

Offset: 11ch l Read/Write: R/W l Reset: 0b01000000

Bit	Reset	Description
7:6	0x1	CFG_RETIME_STAGES
5	0x0	CFG_DEN_EARLY: 0 = DISABLE 1 = ENABLE
4	0x0	CFG_BYPASS_DQS_DLY: 0 = DISABLE 1 = ENABLE
3:0	0x0	CFG_QUSE_LATE

29.4.73 NAND_FBIO_SPARE_0

The FBIO_SPARE register provides extra bits for future needs. They are connected to FBIO at the "common" pad macro, which is near the center of each fbio partition.

FBIO spare register

Offset: 120h l Read/Write: R/W l Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	CFG_FBIO_SPARE

29.4.74 NAND_FC_KNOBS_0

Feature Control register for feature control and perf options enable=1; disable=0 for all features //

Offset: 124h l Read/Write: R/W l Reset: 0b000000000

Bit	Reset	Description
8	0x0	ENABLE_IO_FORCE_FLUSH: Enabling this forces IO state to IDLE immediately in the abort io sequence, Recommended leave default 0 unless used only in conjunction with force abort. 0 = DISABLE 1 = ENABLE
7	0x0	DMA_CLK_GATE_OVERRIDE: Disable 2nd level clock gating for DMA engine. Recommended leave default 0. 0 = DISABLE 1 = ENABLE
6	0x0	ECC_CLK_GATE_OVERRIDE: Disable 2nd level clock gating for ECC engine. Recommended leave default 0. 0 = DISABLE 1 = ENABLE
5	0x0	USE_DOUBLE_BUFFERS_FOR_512B_SECTORS: enabling this turns ON the optimization to use double buffers to fully use all the buffer space when sector size is 512. By default, buffers are 1k sized, so each buffer is used for two 512 sectors. Not supported in Tegra 3. Recommended: leave default 0. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
4	0x0	DISABLE_CONSECUTIVE_REQS: Enabling this disables data requests on consecutive cycles. Recommended setting is 0. 0 = DISABLE 1 = ENABLE
3	0x0	GATE_ABORT_TO_GIZMO: due some possible AHB issue, we may want to gate the abort to the gizmo fifos. Recommended setting 1. 0 = DISABLE 1 = ENABLE
2	0x0	DISABLE_CONTEXT_SWITCH: enabling this turns off the addr/length saves during a main/tag context switch. If this is disabled then main/tag descriptors need to align to page boundaries. i.e. descriptor may not span pages. However, for main_only or tag_only transfers this bit has no effect. Recommended setting is 0. 0 = DISABLE 1 = ENABLE
1	0x0	WORD_ALIGNED_XFER: enabling this makes the DMA bypass the byte reform logic that is needed for byte aligned transfers. Diagnostic workaround for forcing word-aligned transfers. Recommended setting is 0. 0 = DISABLE 1 = ENABLE
0	0x0	PERF_WORD_ALIGNED_XFER: enable this bit when all data transfers (whether in SGDMA mode or not) are guaranteed to be word-aligned. This bit enables an optimization in the request/data queuing which is valid for word-aligned transfers. Recommended setting is 0. 0 = DISABLE 1 = ENABLE

29.4.75 NAND_BUF_STALL_ACCUMULATED_0

Metrics from hardware to enable decision to beef speed_clk

Offset: 128h l Read/Write: R/W l Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	STALL_COUNT

29.4.76 NAND_CORR_FIFO_ACCUMULATED_0

Metrics from hardware to enable decision to beef speed_clk

Offset: 12ch l Read/Write: R/W l Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	EMPTY_COUNT

[THIS PAGE INTENTIONALLY LEFT BLANK]

30.0 GMI CONTROLLER

The Generic Memory Interface (GMI) is a controller that supports a number of memory types, including synchronous NOR, asynchronous NOR, and other flash memories with similar interfaces, such as MuxOneNAND.

The GMI Controller is also known as the Sync-NOR (SNOR) Controller, and enables:

- Data transfer between internal and external memory.
- Interface with Synchronous/Asynchronous Flash Memories.
- Ability to boot from MuxOneNAND.

The GMI/SNOR Controller is on the AHB bus, it provides a way to access external NOR-Flash through Master as well as Slave modes. The SNOR controller supports DMA transfer between Flash Memory and System Memory to save on processor cycles. In order to reduce the number of I/O pins, the NOR interface and NAND interface share pins.

SNOR controller can operate in 2 modes:

- Programmed IO (PIO) mode – In this case the CPU/COP operates as master and issues the read/write commands, and the SNOR controller operates as the slave. The SNOR controller handles the transfer of data between the flash memory and CPU registers or COP
- DMA mode – In this case the SNOR controller operates as the master; the SNOR controller handles the transfer of data between the flash memory and system memory or IRAM

30.1 Functional Description

The GMI/SNOR Controller supports the following operation:

- Synchronous and Asynchronous Flash Memories support
- MuxOneNAND support.
- AHB Master and Slave mode support.
- Booting Option.
- Address-Data Mux / Non-Mux Configuration.
- Fast Access Modes – Burst (Reads/Writes) and Page Mode (Reads).
- Programmable WAIT states for Asynchronous access.
- Programmable Data RDY Configuration (same cycle or next cycle).
- Support 16/32 bits wide Data Bus.
- Support for 8 chip selects: Any combination of 8-SNOR chip selects.

30.1.1 GMI Functional Waveforms

Table 82 GMI (SNOR) Signal Descriptions

Pin Function	Description	Type
Non-Muxed		
GMI_A[27:0]	Address bus: Up to 28 bits of address, depending on type/size of device	Out
GMI_D[31:28] GMI_AD[27:0]	Data bus: 32- or 16-bit data bus	Bidir
Muxed AD		
GMI_D[31:28] GMI_AD[27:0]	Multiplexed Address/Data bus: For 16-bit, muxed A/D memory, the lower 16 bits of address are muxed with the data bus. The upper address lines, GMI_AD[27:16], are non-multiplexed.	Bidir
GMI_A[27:16]	Upper Address bus	Out
General		
GMI_CLK	Clock: Used to synchronize the Tegra® 3 and GMI devices during Synchronous operations. Rising Edge active.	Out
GMI_ADV_N	Address Valid: Programmable polarity output. For synchronous read operations, the address is typically latched either on the inactive edge of ADV_N or on the first rising edge of CLK after ADV_N goes active (slower devices <= 108MHz) or on the last rising edge of CLK after ADV_N goes active (faster devices >= 108MHz) For Asynchronous reads, the address is latched on the inactive of ADV_N. For writes, ADV_N is held active and the address is valid throughout the cycle for non-muxed operation. In the non-muxed case the address will be valid for the duration of the entire access. Only in the muxed mode is it valid during the ADV_N (ADV_WIDTH) + MUXED_WIDTH (the minimum in this case is 1 + 1 = 2 cycles)	Out
GMI_CE[7:0]_N	Chip Enable: Programmable polarity output. When active, CEx_N selects the device and when inactive, de-selects the device which is typically put in low power mode.	Out
GMI_OE_N	Output Enable: Programmable polarity output. When active, OE_N enables the output drivers of the selected (CEx_N active) devices. When inactive, OE_N disables the output drivers of selected devices (CEx_N active) and places them in High-Z.	Out
GMI_WR_N	Write Enable: Programmable polarity output. When active, WR_N enables write operations for the enabled (CEx_N active) devices. The address and write data is latched by the enabled devices on the inactive (trailing) edge of WR_N.	Out
GMI_RST_N	Reset: Active low output. When low, RST_N resets the connected devices and inhibits writes. When high, RST_N enables normal operation.	Out
GMI_WAIT	Wait (or Ready): Level configurable input. When asserted, WAIT (RDY) indicates the read data is invalid (Wait) or Valid (Ready). WAIT (RDY) is driven by the devices when CEx_N and OE_N are low and High-Z when when CEx_N or OE_N are high.	In
GMI_WP_N	Write Protect: Active low output. When low, WP_N enables the device lock down mechanism, or inhibits write cycles.	Out
GMI_DPD	Deep Power Down: Active high output. When high, Device will enter Deep Power Down mode. When low, device will be in or return to normal operating mode.	Out

Figure 78 Synchronous Read (GMI Non-Mux Device)

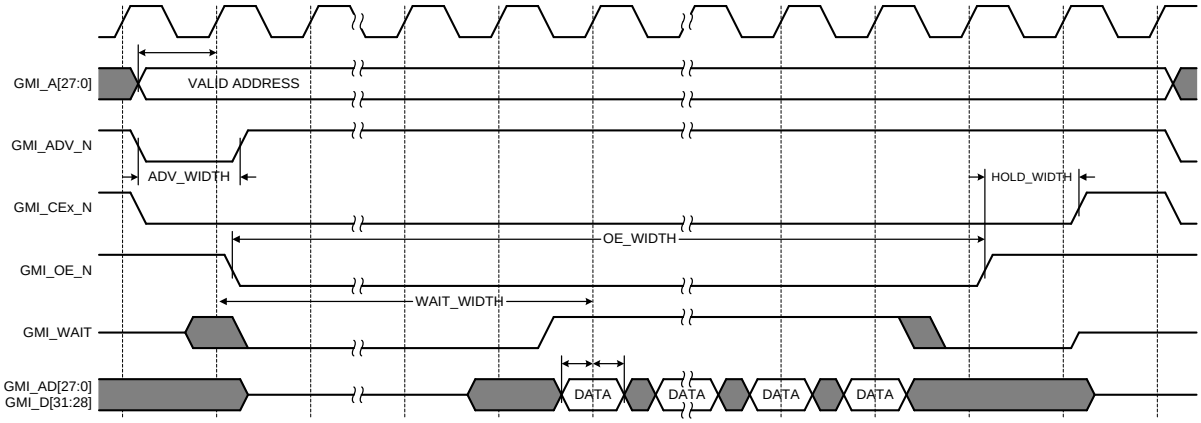


Figure 79 Synchronous Read (GMI Mux-AD 16-bit Data Device)

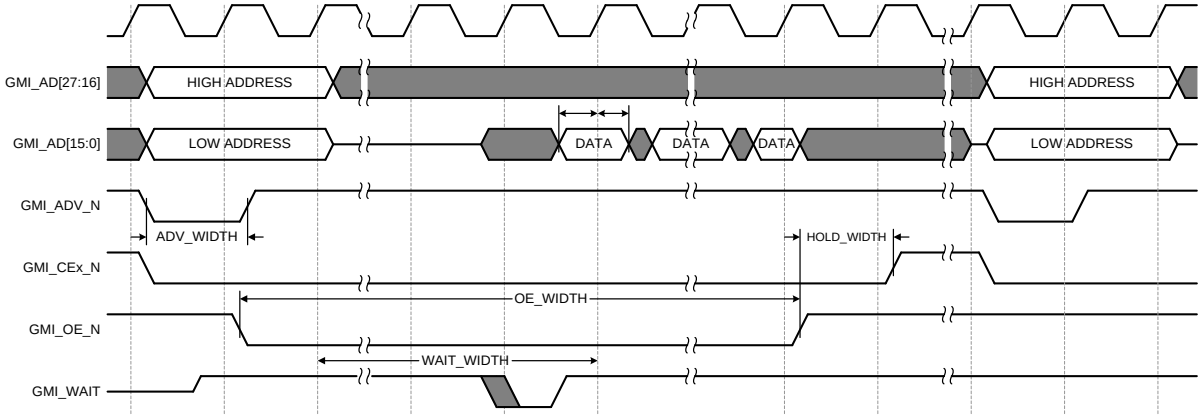


Figure 80 Asynchronous Read (GMI MUX-AD 16-bit Data Device)

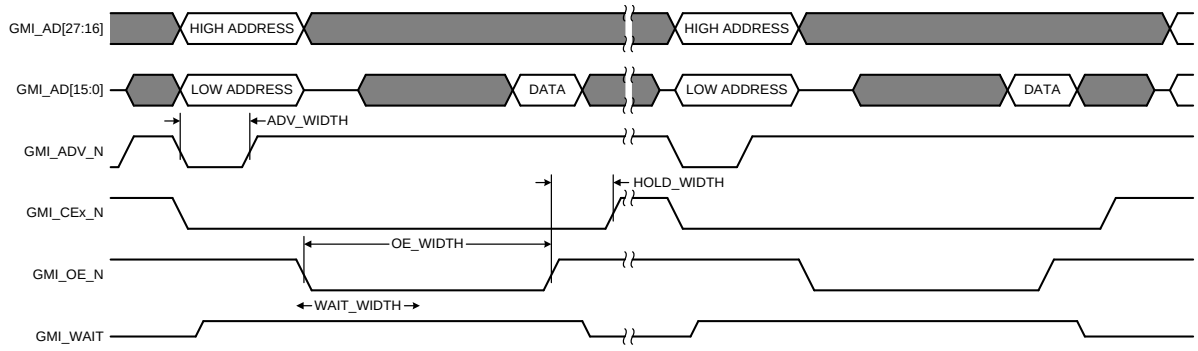
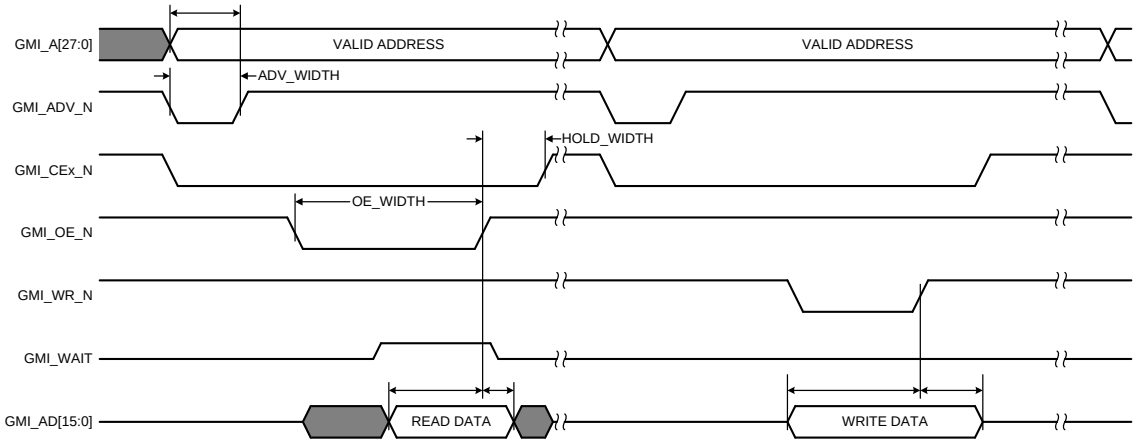


Figure 81 Asynchronous Read Followed by Asynchronous Write (GMI Non-Mux 16-bit Data Device)



30.1.2 GMI Memory Mapping

Table 83 Muxed Memory Mapping

Ball Name	Pin Mux Function	Direction	Type	Pin Function
GMI_CLK	GMI_CLK	O	clock	CLK
GMI_WP_N	GMI_WP_N	O	write protect	WP# / INT
GMI_RST_N	GMI_RST_N	O	reset	RESET#
GMI_DPD	GMI_DPD	O	deep power down	DPD
GMI_CS7_N	GMI_CS7_N	O	chip select	CE#[7]
GMI_CS6_N	GMI_CS6_N	O	chip select	CE#[6]
GMI_CS5_N	GMI_CS5_N	O	chip select	CE#[5]
GMI_CS4_N	GMI_CS4_N	O	chip select	CE#[4]
GMI_CS3_N	GMI_CS3_N	O	chip select	CE#[3]
GMI_CS2_N	GMI_CS2_N	O	chip select	CE#[2]
GMI_CS1_N	GMI_CS1_N	O	chip select	CE#[1]
GMI_CS0_N	GMI_CS0_N	O	chip select	CE#[0]
GMI_OE_N	GMI_OE_N	O	read strobe	OE#
GMI_WR_N	GMI_WR_N	O	write strobe	WE#
GMI_WAIT	GMI_WAIT	I	wait	RDY
GMI_ADV_N	GMI_ADV_N	O	address valid	ADV#
DAP1_SCLK	GMI_D31	B	data	DQ31
DAP1_DOUT	GMI_D30	B	data	DQ30
DAP1_DIN	GMI_D29	B	data	DQ29
DAP1_FS	GMI_D28	B	data	DQ28
GMI_AD27	GMI_AD27	B	address/data	A / DQ27
GMI_AD26	GMI_AD26	B	address/data	A / DQ26
GMI_AD25	GMI_AD25	B	address/data	A / DQ25
GMI_AD24	GMI_AD24	B	address/data	A / DQ24
GMI_AD23	GMI_AD23	B	address/data	A / DQ23
GMI_AD22	GMI_AD22	B	address/data	A / DQ22
GMI_AD21	GMI_AD21	B	address/data	A / DQ21
GMI_AD20	GMI_AD20	B	address/data	A / DQ20
GMI_AD19	GMI_AD19	B	address/data	A / DQ19

Ball Name	Pin Mux Function	Direction	Type	Pin Function
GMI_AD18	GMI_AD18	B	address/data	A / DQ18
GMI_AD17	GMI_AD17	B	address/data	A / DQ17
GMI_AD16	GMI_AD16	B	address/data	A / DQ16
GMI_AD15	GMI_AD15	B	address/data	A / DQ15
GMI_AD14	GMI_AD14	B	address/data	A / DQ14
GMI_AD13	GMI_AD13	B	address/data	A / DQ13
GMI_AD12	GMI_AD12	B	address/data	A / DQ12
GMI_AD11	GMI_AD11	B	address/data	A / DQ11
GMI_AD10	GMI_AD10	B	address/data	A / DQ10
GMI_AD9	GMI_AD9	B	address/data	A / DQ9
GMI_AD8	GMI_AD8	B	address/data	A / DQ8
GMI_AD7	GMI_AD7	B	address/data	A / DQ7
GMI_AD6	GMI_AD6	B	address/data	A / DQ6
GMI_AD5	GMI_AD5	B	address/data	A / DQ5
GMI_AD4	GMI_AD4	B	address/data	A / DQ4
GMI_AD3	GMI_AD3	B	address/data	A / DQ3
GMI_AD2	GMI_AD2	B	address/data	A / DQ2
GMI_AD1	GMI_AD1	B	address/data	A / DQ1
GMI_AD0	GMI_AD0	B	address/data	A / DQ0

Table 84 Non-Muxed Memory Map

Ball Name	Pin Mux Function	Direction	Type	Pin Function
GMI_CLK	GMI_CLK	O	clock	CLK
GMI_WP_N	GMI_WP_N	O	write protect	WP# / INT[1]
GMI_RST_N	GMI_RST_N	O	reset	RESET#
GMI_DPD	GMI_DPD	O	deep power down	DPD
GMI_CS7_N	GMI_CS7_N	O	chip select	CE#[7]
GMI_CS6_N	GMI_CS6_N	O	chip select	CE#[6]
GMI_CS5_N	GMI_CS5_N	O	chip select	CE#[5]
GMI_CS4_N	GMI_CS4_N	O	chip select	CE#[4]
GMI_CS3	GMI_CS3	O	chip select	CE#[3]
GMI_CS2	GMI_CS2	O	chip select	CE#[2]
GMI_CS1	GMI_CS1	O	chip select	CE#[1]
GMI_CS0	GMI_CS0	O	chip select	CE#[0]
GMI_OE_N	GMI_OE_N	O	read strobe	OE#
GMI_WR_N	GMI_WR_N	O	write strobe	WE#
GMI_WAIT	GMI_WAIT	I	wait	RDY
GMI_ADV_N	GMI_ADV_N	O	address valid	ADV#
DAP1_SCLK	GMI_D31	B	data	DQ31
DAP1_DOUT	GMI_D30	B	data	DQ30
DAP1_DIN	GMI_D29	B	data	DQ29
DAP1_FS	GMI_D28	B	data	DQ28
GMI_AD27	GMI_AD27	B	data	DQ27
GMI_AD26	GMI_AD26	B	data	DQ26
GMI_AD25	GMI_AD25	B	data	DQ25
GMI_AD24	GMI_AD24	B	data	DQ24

Ball Name	Pin Mux Function	Direction	Type	Pin Function
GMI_AD23	GMI_AD23	B	data	DQ23
GMI_AD22	GMI_AD22	B	data	DQ22
GMI_AD21	GMI_AD21	B	data	DQ21
GMI_AD20	GMI_AD20	B	data	DQ20
GMI_AD19	GMI_AD19	B	data	DQ19
GMI_AD18	GMI_AD18	B	data	DQ18
GMI_AD17	GMI_AD17	B	data	DQ17
GMI_AD16	GMI_AD16	B	data	DQ16 / INT[2]
GMI_AD15	GMI_AD15	B	data	DQ15
GMI_AD14	GMI_AD14	B	data	DQ14
GMI_AD13	GMI_AD13	B	data	DQ13
GMI_AD12	GMI_AD12	B	data	DQ12
GMI_AD11	GMI_AD11	B	data	DQ11
GMI_AD10	GMI_AD10	B	data	DQ10
GMI_AD9	GMI_AD9	B	data	DQ9
GMI_AD8	GMI_AD8	B	data	DQ8
GMI_AD7	GMI_AD7	B	data	DQ7
GMI_AD6	GMI_AD6	B	data	DQ6
GMI_AD5	GMI_AD5	B	data	DQ5
GMI_AD4	GMI_AD4	B	data	DQ4
GMI_AD3	GMI_AD3	B	data	DQ3
GMI_AD2	GMI_AD2	B	data	DQ2
GMI_AD1	GMI_AD1	B	data	DQ1
GMI_AD0	GMI_AD0	B	data	DQ0
SPI1_CS0_N	GMI_A27	O	address	A27
SPI1_SCK	GMI_A26	O	address	A26
SPI1_MOSI	GMI_A25	O	address	A25
SPI2_CS0_N	GMI_A24	O	address	A24
SPI2_SCK	GMI_A23	O	address	A23
SPI2_MISO	GMI_A22	O	address	A22
SPI2_MOSI	GMI_A21	O	address	A21
DAP2_DOUT	GMI_A20	O	address	A20
DAP2_DIN	GMI_A19	O	address	A19
DAP2_SCLK	GMI_A18	O	address	A18
DAP2_FS	GMI_A17	O	address	A17
DAP4_SCLK	GMI_A16	O	address	A16
DAP4_DOUT	GMI_A15	O	address	A15
DAP4_DIN	GMI_A14	O	address	A14
DAP4_FS	GMI_A13	O	address	A13
GPIO_PU6	GMI_A12	O	address	A12
GPIO_PU5	GMI_A11	O	address	A11
GPIO_PU4	GMI_A10	O	address	A10
GPIO_PU3	GMI_A9	O	address	A9
GPIO_PU2	GMI_A8	O	address	A8
GPIO_PU1	GMI_A7	O	address	A7
GPIO_PU0	GMI_A6	O	address	A6
UART3_CTS_N	GMI_A5	O	address	A5

Ball Name	Pin Mux Function	Direction	Type	Pin Function
UART3_RTS_N	GMI_A4	O	address	A4
UART3_RXD	GMI_A3	O	address	A3
UART3_TXD	GMI_A2	O	address	A2
UART2_CTS_N	GMI_A1	O	address	A1
UART2_RTS_N	GMI_A0	O	address	A0

30.2 Programming Guidelines

During power-on reset the GMI/SNOR Controller has the Slave Interface active. This means any processor can send a request to the GMI Controller/Device. The default setting for the controller as well as external device is in 16-bit MUX mode. Depending on the internal fuses burned, the boot ROM can configure the GMI interface to match that of the external nonvolatile memory, either multiplexed or non-multiplexed. SNOR DMA can be programmed in Burst, Page and Asynchronous modes for NON-MUX devices whereas only Burst and Asynchronous modes are present in case of MUX devices. The GMI Controller supports Burst Writes/Reads, Asynchronous Writes/Reads and Page Reads (only for NON-MUX Devices).

SNOR Register Base address is 0x7000_9000.

SNOR Clock Source Register address is 0x6000_61D0.

The following should be considered while accessing the GMI:

- Using the Slave Interface Asynchronous Writes/Reads are permitted.
- To work in Master Mode, program the “MST_ENB” bit of the Configuration Register.
- Provide the DMA starting NOR Address (0xD000_0000) as well as AHB Address (iRAM 0x4000_0000 or DRAM 0x0000_0000) first, before enabling the “GO_NOR” and “DMA_GO” bits.
- Use the device configuration command cycles to enable the different features of the device including Burst Mode.
- Enable the “GO_NOR” bit followed by “DMA_GO” bit for DMA operation.
- DMA status can be seen by polling the DMA Status Busy Signal or using the transfer done interrupt.
- SNOR timing registers should be programmed per the device specifications.
- In Interrupt polling mode, DMA operation is completed when DMA_GO/GO_NOR are deasserted (disabled).

30.3 GMI Registers

30.3.1 SNOR_CONFIG_0

SNOR Configuration Register

Offset: 000h | Read/Write: R/W | Reset: 0b00010000010000xxx00xxx00000000000

Bit	Reset	Description
31	0x0	GO_NOR: When set a NOR operation commences. 0 = DISABLE 1 = ENABLE
30	0x0	WORDWIDE_GMI: NOR Device DataBus width Configuration Bit. 0 = NOR16BIT 1 = NOR32BIT
29	0x0	NOR_DEVICE_TYPE: External NOR Memory Type. 0 = SNOR 1 = MUXONENAND (simulation purpose)
28	0x1	MUXMODE_GMI: NOR Device Address-Data Configuration Bit. 0 = AD_NONMUX 1 = AD_MUX
27:26	0x0	BURST_LENGTH: Burst Length Types 00=Continuous Burst, 01=8 Words, 10=16 Words, 11=32 Words. 0 = CNTBRST 1 = BL8WORD 2 = BL16WORD 3 = BL32WORD
25	0x0	Reserved
24	0x0	RDY_ACTIVE: Device RDY Active Status 0=With Data, 1=One Cycle Before Data. 0 = WITHDATA 1 = BEFOREDATA
23	0x1	RDY_POLARITY: Ready signal polarity 0=Active low, 1=Active high. 0 = RESV 1 = HIGH
22	0x0	ADV_POLARITY: ADV pulse polarity 0=Active low, 1=Active high. 0 = LOW 1 = RESV
21	0x0	OE_WE_POLARITY: OE/WE polarity 0=Active low, 1=Active high. 0 = LOW 1 = RESV
20	0x0	CS_POLARITY: Chip Select polarity 0=Active low, 1=Active high. 0 = LOW 1 = RESV
19	0x0	NOR_DPD: Indicates the Power Down Mode enable bit. 0 = DISABLE 1 = ENABLE
15	0x0	NOR_WP: Sets the NOR Write protect enable bit. 0 = DISABLE 1 = ENABLE
14	0x0	NMUX_ASYNC_IO: Sets the non-mux async IO mode . 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
10:8	0x0	PAGE_SIZE: Sets the number of words in a page if page mode is selected. 0 = BRST 1 = PG4WORD 2 = PG8WORD 3 = PG16WORD 4 = RESV4 5 = RESV5 6 = RESV6 7 = RESV7
7	0x0	MST_ENB: Selection bit between Master DMA and Slave Interface. 0 = DISABLE 1 = ENABLE
6:4	0x0	SNOR_SEL: SNOR 8 chip selects combinations. 0 = CS0 1 = CS1 2 = CS2 3 = CS3 4 = CS4 5 = CS5 6 = CS6 7 = CS7
3	0x0	CE_LAST: Indicates if the ADV gets asserted before CE. 0 = DISABLE 1 = RESV
2	0x0	CE_FIRST: Indicates if the CE gets asserted before ADV. 0 = DISABLE 1 = RESV
1:0	0x0	DEVICE_MODE: This field specifies the Mode of Operation for SYNC Memories. 0 = ASYNC 1 = PAGE 2 = BURST 3 = RESV

30.3.2 SNOR_STA_0

This status register has bits that provide the controller status, along with the external interrupts from the SNOR devices. The FIFO status is also provided along with the status on the count of the words transferred through DMA

SNOR Status Register

Offset: 004h | Read/Write: R/W | Reset: 0bx0x00000xxxxxxxxxxxxxxxxxxxxxxxx

Bit	R/W	Reset	Description
31	RO	X	DEVICE_BSY: Indicates that the device status.
30	RW	0x0	SLAVE_DONE: Indicates that the slave access is completed
28	RW	0x0	CONT_OVERFLOW: Overflow during continuous mode
27	RW	0x0	DEVICE_INTR_2: Device Interrupt-2 from MuxOneNand Memory.
26	RW	0x0	DEVICE_INTR_1: Device Interrupt-1 from MuxOneNand Memory.
25	RW	0x0	DEVICE_INTR_2_ENB: Device Interrupt-2 Enable Bit.

Bit	R/W	Reset	Description
24	RW	0x0	DEVICE_INTR_1_ENB: Device Interrupt-1 Enable Bit.
23	RO	X	SLV_FIFO_FULL: SLV FIFO full status.
22	RO	X	SLV_FIFO_EMPTY: SLV FIFO empty status.
21	RO	X	MST_FIFO_FULL: MST FIFO full status.
20	RO	X	MST_FIFO_EMPTY: MST FIFO empty status.
15:0	RO	X	DMA_DATA_CNT: Indicates the number of Data to be transferred; current dma_data_count.

30.3.3 SNOR_NOR_ADDR_PTR_0

Contains the starting address of the NOR access.

SNOR Address Register

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SNOR_NOR_ADDR_PTR: Indicates that the NOR controller Address.

30.3.4 SNOR_AHB_ADDR_PTR_0

Contains the starting address of the AHB access.

SNOR AHB Address Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SNOR_AHB_ADDR_PTR: Indicates that the AHB side Address.

30.3.5 SNOR_TIMING0_0

This register contains the timing parameters used for the page access, along with the timing parameters for the widths of ADV pulse, CE hold, etc.

SNOR Timing0 Register

Offset: 010h | Read/Write: R/W | Reset: 0b0011xxxx0001xxxx0001000100010100

Bit	Reset	Description
31:28	0x3	PAGE_RDY_WIDTH: This represents the number of wait clock cycles from address to 1st data ready.
23:20	0x1	PAGE_SEQ_WIDTH: Page Sequential width indicates the delay cycle between the intra page Read access.
15:12	0x1	MUXED_WIDTH: Indicates in number of cycles MUX address/data asserted on the bus.

Bit	Reset	Description
11:8	0x1	HOLD_WIDTH: Indicates in number of cycles CE stays asserted after the de-assertion of WR_N (in case of SLAVE/MASTER Request) or OE_N (in case of MASTER Request).
7:4	0x1	ADV_WIDTH: Indicates the number of cycles during which ADV stays asserted.
3:0	0x4	CE_WIDTH: Indicates the number of cycles before CE is asserted.

30.3.6 SNOR_TIMING1_0

This register contains the timing parameters of the WE and OE widths along with the WAIT width.

SNOR Timing1 Register.

Offset: 014h | Read/Write: R/W | Reset: 0b000000x00000000100000001000000011

Bit	Reset	Description
31:26	0x0	SNOR_TAP_DELAY: Tap delay
24	0x0	SNOR_CE: Clock Override
23:16	0x1	WE_WIDTH: Write access time
15:8	0x1	OE_WIDTH: Read access time.
7:0	0x3	WAIT_WIDTH: Indicates in cycles the number of wait states before when READY is issued.

30.3.7 SNOR_MIO_CFG_0

This register contains the MIO configuration bits such as the width of the words, the polarity of the ready along with the three bits to select the 8 different chip selects for MIO.

MIO Configuration Register

Offset: 018h | Read/Write: R/W | Reset: 0b01xxxxx111

Bit	Reset	Description
29	0x0	MIO_WORDWIDE: Indicates the data bus size of MIO Memory. 0 = MIO16BIT 1 = MIO32BIT
28	0x1	MIO_RDY_POL: Specifies the polarity of MIO RDY. 0 = RESV 1 = HIGH
22:20	0x7	MIO_SEL: MIO 8 chip selects combinations. 0 = MIO0 1 = MIO1 2 = MIO2 3 = MIO3 4 = MIO4 5 = MIO5 6 = MIO6 7 = MIO7

30.3.8 SNOR_MIO_TIMING0_0

This register contains the timing parameters for the MIO accesses MIO Timing - 0 Register

Offset: 01ch | Read/Write: R/W | Reset: 0b000001xx000010xx000001xx000010

Bit	Reset	Description
29:24	0x1	MIO_A_DED_WR: Minimum number of MIO bus clock cycles between the end of a write access and the start of the following access (write or read) for MIO.
21:16	0x2	MIO_A_WR_TIME: Minimum number of MIO bus clock cycles during a write access that the MIO_RD signal is set low for MIO.
13:8	0x1	MIO_A_DED_RD: Minimum number of MIO bus clock cycles between the end of a read access and the start of the following access (write or read) for MIO.
5:0	0x2	MIO_A_RD_TIME: Minimum number of MIO bus clock cycles during a read access that the MIO_RD signal is set low for MIO.

30.3.9 SNOR_DMA_CFG_0

SNOR DMA Configuration Register

Offset: 020h | Read/Write: R/W | Reset: 0b00000100xxxxxxxx00000000000000

Bit	Reset	Description
31	0x0	DMA_GO: This represents the number of DMA is enabled. 0 = DISABLE 1 = ENABLE
30	0x0	BSY: Indicates the status of DMA.
29	0x0	DIR: This represents the direction of DMA data transfer. 0 = NOR2AHB 1 = AHB2NOR
28	0x0	IE_DMA_DONE: Interrupt Enable on DMA transfer completion. 0 = DISABLE 1 = ENABLE
27	0x0	IS_DMA_DONE: Interrupt Status (Write 1 to clear). 0 = DISABLE 1 = ENABLE
26:24	0x4	BURST_SIZE: DMA burst size. 0 = RESV0 1 = RESV1 2 = RESV2 3 = RESV3 4 = BS1WORD 5 = BS4WORD 6 = BS8WORD 7 = RESV7
23	0x0	CONTINUOUS: 0 = Once DMA transfer, 1 = continuous DMA transfer. 0 = DISABLE 1 = ENABLE
15:2	0x0	WORD_COUNT: Specifies the number of words that need to be transferred.

30.3.10 SNOR_CS_MUX_CFG_0

SNOR CHIP Select MUX Configuration Register

Offset: 024h | Read/Write: R/W | Reset: 0b111x110x101x100x011x010x001x000

Bit	Reset	Description
30:28	0x7	CS7_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
26:24	0x6	CS6_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
22:20	0x5	CS5_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
18:16	0x4	CS4_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
14:12	0x3	CS3_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
10:8	0x2	CS2_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
6:4	0x1	CS1_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.
2:0	0x0	CS0_MUX: This represents which chip selects goes to which memory. Chip selection between SNOR and MIO Memories.

[THIS PAGE INTENTIONALLY LEFT BLANK]

31.0 SATA CONTROLLER

31.1 Overview

The Serial Advance Technology Attachment (SATA) controller enables a control path from a Tegra® 3 device to an external SATA device. An SSD/HDD/ODD/e-SATA drive can be connected. This is a one port configuration.

31.2 Device ID

The device ID for the SATA controller in Tegra 3 devices is 0x0E1E. The AHCI driver needs to make sure class code is set to 0x0106 and program interface code set to 0x01 before it starts operating the controller in AHCI mode.

31.3 Power Gating Sequence

This section describes all the steps needed to achieve power gating.

31.3.1 Goals

The power gating POR is:

- Wake events from the drives are supported
- Wake events can only happen in non LP0 mode
- No wake support in LP0 state
- The entire partition including the FPCI wrapper is power gated
- If there can be no wake event, the pads can be put in IDDQ mode. This is decided on the type of drive connected
- The wakeup of the partition (SAX) that hosts the SATA controller and the IPFS shall be initiated by software

31.3.2 Power Gating Overview

Power gating and ungating can be initiated by software. Power gating should be done when the SATA link is in slumber mode, with no outstanding commands. The software should keep a backup of a set of registers in system memory as they lose value once the power is removed. This involves all the configuration space, extended configuration space registers and the MMIO space registers. Since the SW is expected to know the state of registers inside the controller, it need not read the registers every time a power gating sequence is performed.

If HW wake up is to be supported (for e.g., hot plug detection), the pads should be left in slumber. If HW wake up is not needed, then the pads should be placed in the IDDQ mode. This is the preferred mode of operation. The bits pad_iddq_override_en and pad_iddq_override_value could be used to control the state of the pads.

31.3.3 Hardware Initiated Wake up

For a hardware initiated wake up, the pads detect activity on the SATA link. Once the OOB sequence is detected, the bit oob_wake_detected unit is written. Note that the controller itself is not involved in this signal as it is power gated.

The signal from the pads that is of concern here is the rx_stat_idle. This signal must be monitored for any activity. This is an active low signal. There should be two bits that the SW would need to read for the interrupt.

1. To log the status and send an interrupt to SW (SW controlled PG) or signal to the PMC (HW controlled PG). There are three bits associated with this.
    a. A status bit SATA_L0_RX_STAT_IDLE in the register SATA_AUX_RX_STAT_INT_0. This bit shows the current status of rx_stat_idle. It is set to 1 whenever there is activity on the RX path, else it is set to 0.

b. A sticky interrupt status bit SATA_RX_STAT_INT_STATUS in the register SATA_AUX_RX_STAT_INT_0. This is set whenever rx_stat_idle is set to 1 and will stay 1 until the interrupt is cleared by SW.

c. There are bits to set/clear the bits above. Writing a 1 to SATA_RX_STAT_INT_SET bit in SATA_AUX_RX_STAT_SET_0 register will set the status. Writing a 1 to SATA_RX_STAT_INT_CLR will clear this status.

2. To have a mask to enable / disable interrupts. This is done via central interrupt controller with the interrupt SATA_RX_STAT in PRI_ICTLR_* registers. Follow the same guidelines as any other units to enable/disable the SATA_RX_STAT interrupt.

The SW clears the interrupt and the mask during normal operation.

31.3.4 Bits Required in Power Gating Sequences

The following bits are required in power gating sequences. Some of them are needed in both HW and SW sequencing methods. Some of them are required only in one of the modes of power gating. The bits could be named differently. The bits are –

- Oob_wake_en – this bit is to be placed in a non power gated region. Its role is to enable wake events from the drive. This is enabled by writing a 1 to SATA_RX_STAT in PRI_ICTLR_* registers
- Oob_wake_detected – this bit shall be written when “oob_wake_en” is asserted and a wake is detected on the SATA link. The “rx_stat_idle” toggling shall assert this bit. In SW mode of power gating, an interrupt needs to be sent to the SATA driver. This is the SATA_RX_STAT_INT_STATUS bit in register SATA_AUX_RX_STAT_INT_0.
- Pmc2sata_pg_info – this bit shall drive the signal pmc2sata_pg_info.

31.3.5 SW Initiated Power Gating / un-Gating Sequence

Steps involved in Power Gating

#	Description	Register Programming	Comments
1.	Driver decides to enter power gating, based on the fact that links are in slumber state and no commands are outstanding.		
2.	Driver programs a register in the PMC to indicate to SATA that it is entering power gating. This shall drive the pmc2sata_pg_info signal.	APBDEV_PMC_SATA_PWRGT_0.PG_INFO = 1	If driver detects a HW wake from the drive before setting PG_INFO bit, it should bail out and should not begin the PG sequence.
3.	Driver would read out the registers in the SATA controller and save it in system memory.	Register list is in appendix.	The time taken for this would depend on the time taken for the SW to read a register. Ideally this should be done only once.
4.	Driver needs to read and set the rx_idle_t registers in apbmisc so idle threshold can be continuously driven while SAX is power-gated	Read SATA_AUX_MISC_CNTL_1_0 register L0_RX_IDLE_T_SAX field and write that value into same register L0_RX_IDLE_T_NPG field. And write ‘1’ to L0_RX_IDLE_T_MUX field.	
5.	Driver should read the error and interrupt registers before entering power gating.	Read PxSERR and PxIS, and IS registers. Also read PxCI register to make sure that there are no outstanding commands. If (PxCI !=0) or any unexpected error, driver should bail out of PG sequence by setting APBDEV_PMC_SATA_PWRGT_0.PG_INFO = 0 Sets the bit rx_stat_idle_bypass to enable detection of a comwake.	If there is something unexpected or an error is detected or if HW wake is detected from the drive, then the SW should bail out of PG sequence (clear PMC bit) Once PG_INFO=1 and/or rx_stat_idle bypass is enabled, HW wake could be detected via rx_stat_idle interrupt only. In that case, driver needs to wait until HBA is in active state (PxSSTS.DET=0x3 and PxSSTS.IPM=0x1) before issuing any new commands.

#	Description	Register Programming	Comments
6.	Driver should gate SATA Clocks and assert reset to SATA.	CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0 .SATA_CLKEN= '0' CLK_RST_CONTROLLER_RST_DEVICES_V_0. SATA_RESET=1 CLK_RST_CONTROLLER_RST_DEVICES_W_0 .SATACOLD_RST=1	
7.	*If HW wake up (example hot-plug or async notification) is needed:* Driver shall write a bit in Intr Ctlr in non-power gated region to enable the rx_stat interrupt generation on detection of rx_stat(OOB) wake.	Clear sata_rx_stat interrupt status and enable sata_rx_stat interrupt to start detecting rx_stat_idle changes from sata pad when detecting oob wake sequence : Write apb_misc sata_aux register RX_STAT_CLR bit[0] "SATA_RX_STAT_INT_CLR" field to '1' Write interrupt controller register **PRI_ICTLR_CPU_IER_SET_0 bit[13] "**SATA_RX_STAT" field to '1' The mapped status bit for sata_rx_stat interrupt in interrupt controller is at register **PRI_ICTLR_ISR_0 bit[13] "SATA_RX_STAT"**	At this point, sideband(phy logic + intr ctlr) takes over interrupt generation.
8.	*If HW wake up (example hot-plug or async notification) is needed:* Driver/RM shall place the SATA PADPLL in reset. *If Hw wake up is not needed:* Driver/RM shall place the SATA PHY and SATA PADPLL in IDDQ.	a) SATA_PADPLL_RESET_SWCTL =1 SATA_PADPLL_RESET_OVERRIDE_VALUE=1 b) SATA_PADPLL_RESET_SWCTL =1 SATA_PADPLL_RESET_OVERRIDE_VALUE=1 SATA_PADPHY_IDDQ_SWCTL=1 SATA_PADPHY_IDDQ_OVERRIDE_VALUE=1 Wait for time specified in SATA_LANE_IDDQ2_PADPLL_IDDQ SATA_PADPLL_IDDQ_SWCTL=1 SATA_PADPLL_IDDQ_OVERRIDE_VALUE=1	Whether this is done by driver or RM depends on whether we implement HW low power sequencing. Guideline: if SATA PLL is to be controlled by SW at run time (no HW low power sequencing), then it should be done by driver, not RM. Conversely, if it is to be controlled by SW only during power gating, then it can be in RM.
9.	Driver reports to the RM that SATA is being powered down. If PCIe® is not being used, RM can set the PLLE to either reset or IDDQ. *If HW wake up is needed and exit latency cannot be met with PLLE IDDQ mode, then* PLLE can be set to reset. *If HW wake up is not needed or PLLE IDDQ mode exit latency is sufficient:* PLLE can be set to IDDQ.	a) PLLE_ENABLE_SWCTL=1 CLK_RST_CONTROLLER_PLLE_BASE_0.PLLE _ENABLE=0 b) PLLE_ENABLE_SWCTL=1 CLK_RST_CONTROLLER_PLLE_BASE_0.PLLE _ENABLE=0 PLLE_IDDQ_SWCTL=1 PLLE_IDDQ_OVERRIDE_VALUE=1	RM takes over from this point of time. Wait Time= 1us?
10.	RM now shall power gate the controller. This shall also involve the setting of the clamps, and assertion of the resets.	APBDEV_PMC_PWRGATE_TOGGLE_0.SAX=1 << reset overrides exist in the CAR block >>	The clamp value of the HW is such that it shall place the SATA PAD PLL in IDDQ mode but it is overridden by sw value as swctl on iddq/reset is enabled above.
11.	RM indicates to the AHCI SW, that power gating is complete.		This is more of an implicit step; returning of the RM PG API function is considered RM telling module driver that PG is done.

Steps involved in Power Ungating

#	Description	Register Programming	Comments
1	If HW wake up is allowed and an OOB sequence is detected, an rx_stat interrupt will be sent by the interrupt ctlr Or, driver decides to wake up the controller.	When sata_rx_stat interrupt is detected and serviced, interrupt routine should disable this interrupt and clear the status by doing : Write interrupt controller register PRI_ICTLR_CPU_IER_CLR_0 bit[13] "SATA_RX_STAT" field to '1' Write apb_misc sata_aux register RX_STAT_CLR	

#	Description	Register Programming	Comments
		bit[0] “SATA_RX_STAT_INT_CLR” field to ‘1’	
2	If SATA PHY and SATA PADPLL is in IDDQ mode, driver/RM brings it out of IDDQ mode.	Get SATA PHY and PLL out of IDDQ as below: SATA_PADPLL_IDDQ_OVERRIDE_VALUE=0 Wait for programmable delay of SATA_PADPLL_IDDQ2LANE_SLUMBER_DLY. Suggested value of this delay is 3 microseconds. SATAPAD_IDDQ_OVERRIDE_VALUE=0	
3	RM checks if PLLE is already turned on or not. Part (a) : If PLLE is already turned on, it waits for TBD microseconds before moving on to next step. Part (b): If PLLE is off, it brings it up.	If PCIe is running, PLLE could be already running. Check that it is running by checking its PLLE_ENABLE and PLLE_LOCK bit. Part (a): If they both are set, then PLLE is already turned on. Wait for TBD microseconds. Part (b) If they both aren’t set, then do PLLE training and init sequence. If using spread spectrum, set the following bits: CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCBYP=1 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_BYPASS_SS=1 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_INTERP_RESET=1 Set PLLE_LOCK_ENABLE bit in CLK_RST_CONTROLLER_PLLE_MISC register to DISABLE (0). Set the PLLE_ENABLE and PLLE_ENABLE_CML bits to DISABLE (0) in CLK_RST_CONTROLLER_PLLE_BASE register. Also set PLLE_SETUP=0 and PLLE_EXT_SETUP_17_16=0 in CLK_RST_CONTROLLER_PLLE_MISC register. Check that PLLE_IDDQ_SWCTL=1 and PLLE_IDDQ_OVERRIDE_VALUE=1. If they are not set to 1, set them to 1. Then set PLLE_IDDQ_OVERRIDE_VALUE=0 to begin the training sequence. Wait for PLLE training sequence to finish by waiting until PLLE_PLL_READY bit is set to 1 in CLK_RST_CONTROLLER_PLLE_MISC register program PLLE parameters (most are the default POR values) – set the following values: CLK_RST_CONTROLLER_PLLE_BASE register: PLLE_PLDIV_CML = 0xd PLLE_PLDIV = 0x18 PLLE_MDIV = 0x1 PLLE_NDIV = 0xc8 I*. These settings are for Tegra 3: PLLE_PLDIV_CML = 0xb PLLE_PLDIV = 0x12 PLLE_MDIV = 0x1 PLLE_NDIV = 0x96 CLK_RST_CONTROLLER_PLLE_MISC register PLLE_SETUP = 0x7 (This isn’t default) PLLE_LOCK_ENABLE = ENABLE (1) (This isn’t default) If using spread spectrum, set the following bits:	

#	Description	Register Programming	Comments
		CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCBYP=1 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_BYPASS_SS=1 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_INTERP_RESET=1 Enable the PLLE now by setting PLLE_ENABLE and PLLE_ENABLE_CML to ENABLE (1) in CLK_RST_CONTROLLER_PLLE_BASE register. Wait for PLLE to lock by waiting until PLLE_LOCK bit is set to 1 in CLK_RST_CONTROLLER_PLLE_MISC register. After PLLE_LOCK is detected, wait for an additional programmable delay of PLLE_LOCK_DLY. This will help in case the PLLE_LOCK gets asserted earlier. The suggested value of this additional delay is 25 microseconds. Do the following steps if you intend to use spread spectrum: program spread spectrum coefficients- CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCMAX = 0x29, CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCINCINTRV = 0x1d, CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCINC = 0x1. 11*) These settings are for Tegra 3: CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCMAX = 0x24, CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCINCINTRV = 0x18, CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCINC = 0x1. CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_BYPASS_SS=0 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_SSCBYP=0 CLK_RST_CONTROLLER_PLLE_SS_CNTL_0.PLLE_INTERP_RESET=0 Enable CML clock for SATA by setting PLLE_CML1_OEN bit to 1 in CLK_RST_CONTROLLER_PLLE_AUX_0 register. If PLLE is already turned on, check that the SATA CML clock output is turned on. Otherwise, turn it on by setting PLLE_CML1_OEN bit to 1 in CLK_RST_CONTROLLER_PLLE_AUX_0 register.	
4	RM deasserts reset to SATA PADPLL and waits until it locks.	SATA_PADPLL_RESET_OVERRIDE_VALUE=0 Wait for Satapll_lockdet =’1’	Expected maximum time for SATA PADPLL to lock = 15 microseconds.
5	RM toggles the powergate sax bit in PMC to power-ungate SAX partition.	Toggle the SAX partition register bits in PMC APBDEV_PMC_PWRGATE_TOGGLE_0.SAX=0 APBDEV_PMC_REMOVE_CLAMPING_CMD_0.SAX=1	
6	RM returns the status to the driver that the power un-gating is complete.		This is an implicit step. No action here.
7	Driver turns on the clocks to SATA and deasserts resets.	SW to Ungate txclk and all SATA clks, reset CLK_RST_CONTROLLER_CLK_OUT_ENB_V_0.SATA_CLKEN= ‘1’ ,	

#	Description	Register Programming	Comments
		CLK_RST_CONTROLLER_RST_DEVICES_W_0 .SATACOLD_RST= ‘0’ , CLK_RST_CONTROLLER_RST_DEVICES_V_0. SATA_RESET =0	
8	Driver restores the registers of the controller.	Driver should restore the registers in the following order: IPFS, CFG, Ext CFG, BAR5. Make sure that any registers that change values based on customer design are restored to their proper values. During the restoration of the registers, the driver would now need to restore the register T_SATA0_CFG_POWER_GATE_SSTS_RESTORED after the ssts_det, ssts_spd are restored. This register is used to tell the controller whether a drive existed earlier or not and move the PHY state machines into either HR_slumber or not.	Expected to take ~10 us.
9	Driver needs to switch the rx_idle_t driven source back to from SATA controller after SAX is power-ungated	Write SATA_AUX_MISC_CNTL_1_0 register L0_RX_IDLE_T_MUX field to ‘0’.	
10	Driver can start to use main SATA interrupt instead of the rx_stat_t interrupt.	After SATA logic power-ungated we can start to use main interrupt from sata controller and interrupt status mapped to interrupt controller register PRI_ICTLR_ISR_0 bit[23] “SATA_CTL”	
11	Set the bits in the CAR to allow HW based low power sequencing.	SATAPAD_PLL_PLLE_IDDQ_RESET_SWCTL =0	
12	Driver indicates to controller that power un-gating process is complete.	Clear bit in PMC . APBDEV_PMC_SATA_PWRGT_0.Pmc2sata_pg_info = 0 Clears the bit rx_stat_idle_bypass to enable detection of a comwake.	

Note: Wakeup behavior when CPU is off

Wake up might include waking the CPU itself

If the CPU is powered down, the rx_stat interrupt causes flow controller/PMC to power up CPU, CPU goes to reset vector handler code in OS kernel patch which will restore CPU state, reinit CPU, un-powergate SATA, restore SATA state, gives control to AHCI SW.

If the CPU is powered up, the rx_stat interrupt signal generates a CPU interrupt, interrupt handler in driver shall un-powergate SATA, restore SATA state.

31.3.6 Software Changes

In SW initiated power gating and ungating, the entry and exit of the power gating the controller is controlled by the software. In the HW initiated wake up mode, the software would need to read the bit oob_wake_detected to determine if the controller needs to be power ungated. The software would need to keep a backup of all the registers in system memory and would need to reprogram them once in the power un-gating process. The software would need to program the registers in the correct sequence as mentioned above.

There is a bug in the hardware, where the PxCMD.CR does not have the right value after register restoration following power-ungating. The Software should not read that bit.

A software workaround is needed when SW is initiating power-gating. To prevent an accidental COMRESET from being sent on the SATA link, software will need to set override for HW to not look at SATA PLL lockdet when shutting down SATA PLL.

This can be done by writing SATA ext. config space register T_SATA0_CFG_MISC(0x550), need to write bit[10] to ‘1’ and bit[8] to ‘0’ (field T_SATA0_CFG__PHY_RESET_USAGE_MODE).

31.3.7 Registers to Save and Restore

This list of registers that require save and restore across power gating sequence is included below. Apart from this the software is expected to save and restore the configuration space registers.

Save and restore

T_AHCI_HBA_SPARE_0
T_AHCI_HBA_GHC
T_AHCI_HBA_EM_LOC
T_AHCI_PORT_PXCLB
T_AHCI_PORT_PXFB
T_AHCI_PORT_PXIE
T_AHCI_HBA_SHUTDOWN_TIMER
T_AHCI_PORT_PXSCTL
T_AHCI_PORT_MP
T_AHCI_HBA_CCC_PORTS
T_AHCI_HBA_CCC_CTL - OP (optional)
T_AHCI_HBA_EM_CTL - OP
T_AHCI_PORT_PXCLBU
T_AHCI_PORT_PXFBU
T_AHCI_PORT_PXCMD
T_AHCI_HBA_PLL_CTRL
T_AHCI_PORT_PXFBS

Save and restore via bkdr writes

T_AHCI_HBA_CAP
T_AHCI_HBA_CAP2
NV_PROJ__SATA0_CHX_AHCI_PORT_PXSIG
T_AHCI_PORT_BKDR
T_AHCI_HBA_PI
NV_PROJ__SATA0_CHX_AHCI_PORT_PXTFD
NV_PROJ__SATA0_CHX_AHCI_PORT_PXSSTS

Registers to read only, not needed to restore

T_AHCI_HBA_BOHC
T_AHCI_PORT_INTR
T_AHCI_PORT_PXSERR

31.4 Address Translation

The bar spaces and PCI™ Config space from the PCI bus environment are mapped as is to an address space carved out of ARM (ordering of the registers is not changed). Tegra 3 devices do not follow the PCI specifications. They do not have any concept of the PCI configuration space, BAR space, and the extended configuration space. The system addressing is of 32 bits, unlike 40 bits in the PCI bus environment. The address can map up to 4 GB.

There is an address translation mechanism in the IPFS. With this address translation, a certain address range is mapped to the configuration space registers, and likewise for the BAR 5 and the extended configuration space registers.

The address mapping that is being done is explained in the table in the Appendix. Even though, BAR 0 to Bar 4 are not applicable to Tegra 3 devices, they too have been allocated address ranges.

31.4.1 Address Translation Table

The next table shows how all of the SATA register spaces and SATA-IPFS registers are mapped to Tegra 3 memory map. In Tegra 3, 64KB APB space is allocated for all these registers and 512B APB space allocated for SATA iobist registers.

Register Section		DFPCI address [39:0]	APB adr[31:0] for DFPCI mapping	PRI address [31:0]	APB adr[31:0] for PRI mapping
			*APB address 0x7002_0000 - 0x7002_0FFF is reserved for IPFS local registers		
1. SATA configuration space		Address starting with 0xFD_FE or 0xFE_0			
SATA configuration space 0x0 - 0x255	Start address	0xFD_FE00_5000	0x7002_1000	0x5010_0000	0x7002_9000
	End address	0xFD_FE00_50FF	0x7002_10FF	0x5010_00FF	0x7002_90FF
SATA extended configuration space 0x100 - 0xFFF	Start address	0xFE_0100_5000	0x7002_1100	0x5010_0100	0x7002_9100
	End address	0xFE_0F00_50FF	0x7002_1FFF	0x5010_0FFF	0x7002_9FFF
		Format : 0xFE_0X00_50YZ			
		XYZ from 0x100 to 0xFFF			
I/O space		Address starting with 0xFD_FC			
2. SATA BAR0					
Register MCP_SATA_PRI_COMMAND_0	Address in SATA Compatible mode	0xFD_FC00_01F0	0x7002_21F0	0x5020_0200	0x7002_A200
	Address in SATA Native mode	SATA BAR0 with offset 0x0	0x7002_2000	0x5000_0200	0x7002_A200
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_2000	Not used but address can be same as native mode	0x7000_A200
Register MCP_SATA_PRI_COMMAND_1	Address in SATA Compatible mode	0xFD_FC00_01F4	0x7002_21F4	0x5020_0204	0x7002_A204
	Address in SATA Native mode	SATA BAR0 with offset 0x4	0x7002_2004	0x5000_0204	0x7002_A204
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_2004	Not used but address can be same as native mode	0x7000_A204
3. SATA BAR1					
Register MCP_SATA_PRI_CONTROL	Address in SATA Compatible mode	0xFD_FC00_03F4	0x7002_33F4	0x5020_0300	0x7002_A300
	Address in SATA Native mode	SATA BAR1 with offset 0x0	0x7002_3000	0x5020_0300	0x7002_A300
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_3000	Not used but address can be same as native mode	0x7002_A300

Register Section		DFPCI address [39:0]	APB adr[31:0] for DFPCI mapping	PRI address [31:0]	APB adr[31:0] for PRI mapping
4. SATA BAR2					
Register MCP_SATA_SEC_COMMAND_0	Address in SATA Compatible mode	0xFD_FC00_0170	0x7002_4170	0x5020_0400	0x7002_A400
	Address in SATA Native mode	SATA BAR2 with offset 0x0	0x7002_4000	0x5020_0400	0x7002_A400
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_4000	Not used but address can be same as native mode	0x7002_A400
Register MCP_SATA_SEC_COMMAND_1	Address in SATA Compatible mode	0xFD_FC00_0174	0x7002_4174	0x5020_0404	0x7002_A404
	Address in SATA Native mode	SATA BAR2 with offset 0x4	0x7002_4004	0x5020_0404	0x7002_A404
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_4004	Not used but address can be same as native mode	0x7002_A404
5. SATA BAR3					
Register MCP_SATA_SEC_CONTROL	Address in SATA Compatible mode	0xFD_FC00_0374	0x7002_5374	0x5020_0500	0x7002_A500
	Address in SATA Native mode	SATA BAR3 with offset 0x0	0x7002_5000	0x5020_0500	0x7002_A500
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_5000	Not used but address can be same as native mode	0x7002_A500
6. SATA BAR4					
Bus Master Registers 0x0 - 0xF	Address in SATA Compatible mode	SATA BAR4 with offset 0x0 - 0xF	0x7002_6000 - 0x7002_600F	0x5020_0600 - 0x5020_060F	0x7002_A600 - 0x7002_A60F
	Address in SATA Native mode	SATA BAR4 with offset 0x0 - 0xF	0x7002_6000 - 0x7002_600F	0x5020_0600 - 0x5020_060F	0x7002_A600 - 0x7002_A60F
	Address in SATA AHCI mode	Not used but address can be same as native mode	0x7002_6000 - 0x7002_600F	Not used but address can be same as native mode	0x7002_A600 - 0x7002_A60F
Memory space		Address[39:32] = 0x0			
7. SATA BAR5 (claim 8KB)					
AHCI Registers 0x0 - 0x1FFF	Start address	SATA BAR5 with offset 0x0	0x7002_7000	0x5030_0000	0x7002_B000
	End address	SATA BAR5 with offset 0x1FFF	0x7002_8FFF	0x5030_1FFF	0x7002_CFFF
8. SATA IOBIST registers	Start address	Non accessible	N.A.	0x5050_0000	0x7002_D000
	End address	Non accessible	N.A.	0x5050_01F7	0x7002_D1F7
Note : In Tegra 3, there is a standalone APB client created for SATA iobist register access, the PRI interface mapping from SATA APB client is no longer valid					
The SATA IOBIST registers are now mapped to 0x7000_6C00 - 0x7000_6DFF					

31.5 Registers for Save and Restore

The registers needed to be saved and restored are

IPFS registers

Save and restore	Offset	Save and restore	Offset
SATA_FPCI_BAR5_0	94	SATA_MSI_EN_VEC6_0	158
SATA_MSI_BAR_SZ_0	C0	SATA_MSI_EN_VEC7_0	15C
SATA_MSI_AXI_BAR_ST_0	C4	SATA_CONFIGURATION_0	180
SATA_MSI_FPCI_BAR_ST_0	C8	SATA_FPCI_ERROR_MASKS_0	184
SATA_MSI_EN_VEC0_0	140	SATA_INTR_MASK_0	188
SATA_MSI_EN_VEC1_0	144	SATA_IPFS_INTR_ENABLE_0	198
SATA_MSI_EN_VEC2_0	148	SATA_CFG_REVID_0	1A0
SATA_MSI_EN_VEC3_0	14C	SATA_CLKGATE_HYSTERSIS_0	1BC
SATA_MSI_EN_VEC4_0	150	SATA_SATA_MCCIF_FIFOCTRL_0	1DC
SATA_MSI_EN_VEC5_0	154		

Configuration Space and Extended Configuration Space Registers

Save and restore	offset	Save and restore	offset
T_SATA0_CFG_1	4	T_SATA0_CFG_PHY_1	12C
T_SATA0_CFG_3	C	T_SATA0_CFG_LINK_0	174
T_SATA0_CFG_9	24	T_SATA0_CFG_LINK_1	178
T_SATA0_CFG_10	28	MCP_SATA0_CFG_TRANS_0	1D0
T_SATA0_CFG_12	30	T_SATA0_ALPM_CTRL	238
T_SATA0_CFG_13	34	T_SATA0_AHCI_HBA_CYA_0	30C
T_SATA0_CFG_14	38	T_SATA0_AHCI_HBA_BIST_OVERRIDE_CTL	318
T_SATA0_CFG_15	3C	T_SATA0_AHCI_HBA_SPARE_1	320
T_SATA0_CFG_16	40	T_SATA0_AHCI_HBA_SPARE_2	324
T_SATA0_CFG_17	44	T_SATA0_AHCI_HBA_DYN_CLK_CLAMP	328
T_SATA0_CFG_18	48	T_SATA0_AHCI_CFG_ERR_CTRL	32C
T_SATA0_MSI_CTRL	B0	T_SATA0_AHCI_HBA_CYA_1	338
T_SATA0_MSI_ADDR1	B4	T_SATA0_AHCI_HBA_PRE_STAGING_CONTROL	340
T_SATA0_MSI_ADDR2	B8	T_SATA0_CFG_FPCI_0	430
T_SATA0_MSI_DATA	BC	T_SATA0_CFG_ESATA_CTRL	494
T_SATA0_MSI_QUEUE	C0	T_SATA0_CYA1	4A0
T_SATA0_MSI_MAP	EC	T_SATA0_CFG_GLUE	4B0
T_SATA0_CTRL	540	T_SATA0_PHY_CTRL	534
T_SATA0_CFG_PHY_POWER	124	T_SATA0_CTRL	540
T_SATA0_CFG_PHY_POWER_1	128	T_SATA0_LOW_POWER_COUNT	554

Save and Restore Following Protocol

(Program T_SATA0_INDEX to select port 0)

Save and restore	offset	Save and restore	offset
T_SATA0_CHXCFG1	530	T_SATA0_CHX_PHY_CTRL_3	6B0
T_SATA0_CHX_MISC	684	T_SATA0_CHX_PHY_CTRL_4	6B4
T_SATA0_CHXCFG3	700	T_SATA0_CHX_PHY_CTRL_5	6B8

Save and restore	offset	Save and restore	offset
T_SATA0_CHXCFG4_CHX	704	T_SATA0_CHX_PHY_CTRL_6	6BC
T_SATA0_CHX_PHY_CTRL1_GEN1	690	T_SATA0_PRBS_CHX – OP	714
T_SATA0_CHX_PHY_CTRL1_GEN2	694	T_SATA0_CHX_LINK0	750
T_SATA0_CHX_PHY_CTRL1_GEN3	698	T_SATA0_CHX_GLUE	7F0
T_SATA0_CHX_PHY_CTRL_2	69C		

BAR 5 Space Registers

Save and restore	offset	Save and restore	offset
T_AHCI_HBA_CCC_PORTS	18	T_AHCI_PORT_PXCLBU	104
T_AHCI_HBA_GHC	4	T_AHCI_PORT_PXFB	108
T_AHCI_HBA_CCC_CTL - OP (optional)	14	T_AHCI_PORT_PXFBU	10C
T_AHCI_HBA_EM_LOC	1C	T_AHCI_PORT_PXIE	114
T_AHCI_HBA_EM_CTL - OP	20	T_AHCI_PORT_PXCMD	118
T_AHCI_PORT_PXCLB	100	T_AHCI_PORT_PXSCTL	12C

Save and Restore via bkdr Writes

Save and restore	offset
T_AHCI_HBA_CAP	0 300*
T_AHCI_HBA_PI	0C 33C*
T_AHCI_HBA_CAP2	24 330*
NV_PROJ__SATA0_CHX_AHCI_PORT_PXTFD	120 790*
NV_PROJ__SATA0_CHX_AHCI_PORT_PXSIG	124 794*
NV_PROJ__SATA0_CHX_AHCI_PORT_PXSSTS	128 798*

Note: * Backdoor adresses for restore

Registers to Read Only (Not Needed to Restore)

T_AHCI_HBA_BOHC 28
T_AHCI_PORT_INTR 174
T_AHCI_PORT_PXSERR 130

31.6 Registers Used in Power Gating/Un-gating Sequence

31.6.1 APBDEV_PMC_SATA_PWRGT_0

This register is also a part of the PMC section of this document.

IDDQ override Regs (bits 5:0)

Offset: 1ach | Read/Write: R/W | Reset: 0b00111111

Bit	Reset	Description
7	0x0	SW_PLL_EDPD: sata pll put in dpd state when bit is set, independently of powergating - 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 Sw can put the plle in iddq

Bit	Reset	Description
		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 HW (SATA +AFI+CAR) signals. 1 The PLLE is put in IDDQ by SW -- default 0 = OFF 1 = ON
3	0x1	PADPHY_IDDQ_OVERRIDE_VALUE: 0 The PHY is powered up 1 Sw can put the PHY in IDDQ 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 HW (SATA +AFI+CAR) signals. 1 The Sataphy is put in IDDQ by SW - default 0 = OFF 1 = ON
1	0x1	PADPLL_IDDQ_OVERRIDE_VALUE: 0 The pad PLL is powered up 1 SW can put the pad PLL in IDDQ by setting this bit and SATA_PADPLL_IDDQ_SWCTL- default 0 = OFF 1 = ON
0	0x1	PADPLL_IDDQ_SWCTL: 0 The Sata padpll is put in IDDQ mode by HW (SATA +AFI+CAR) signals.1 The SATA padpll is put in IDDQ by SW. - default 0 = OFF 1 = ON

31.6.2 CLK_RST_CONTROLLER_SATA_PLL_CFG0_0

This register is also a part of the CAR section of this document.

SATA power sequencer input SW control program guide

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

SATA power sequencer is driven by three primary reset and powerdown input signals: reset, lane_pd, padpll_pd.

Normal power up sequence is to deassert in the order of padpll_pd->lane_pd->reset.

Normal power down sequence is to assert in the order of reset->lane_pd->padpll_pd.

It is acceptable to assert all three at the same time.

It is acceptable for reset or lane_pd to return to power-up state in the middle of a power-down sequence.

SATA PLL Reset Override Register (bit 1:0)

Offset: 490h | Read/Write: R/W | Reset: 0bxx10xxxxxxxxxxxxxxx0000x011

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

Bit	R/W	Reset	Description
7	RW	0x0	SATA_SEQ_PADPLL_PD_INPUT_VALUE: 0=pad PLL is power up, 1=SW control padpll PD by setting this and SATA_SEQ_IN_SWCTL bit
6	RW	0x0	SATA_SEQ_LANE_PD_INPUT_VALUE: 0=IOPHY is power up, 1=SW control IOPHY PD by setting this and SATA_SEQ_IN_SWCTL bit
5	RW	0x0	SATA_SEQ_RESET_INPUT_VALUE: 0=SATA reset deasserted, 1=SW control reset by setting this and SATA_SEQ_IN_SWCTL bit
4	RW	0x0	SATA_SEQ_IN_SWCTL: 0=seq input by HW, 1=seq input by SW
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 power up, 1=SW control reset by setting this and SATA_PADPLL_RESET_SWCTL bit
0	RW	0x1	SATA_PADPLL_RESET_SWCTL: 0=reset by HW, 1=reset by SW

31.6.3 CLK_RST_CONTROLLER_PLLE_AUX_0

This register is also a part of the CAR section of this document.

Note that Range value is the recommended range. All counters supports 0-255 step range.

Offset: 48ch | Read/Write: R/W | Reset: 0b0xxxxx10000001000100000001110000

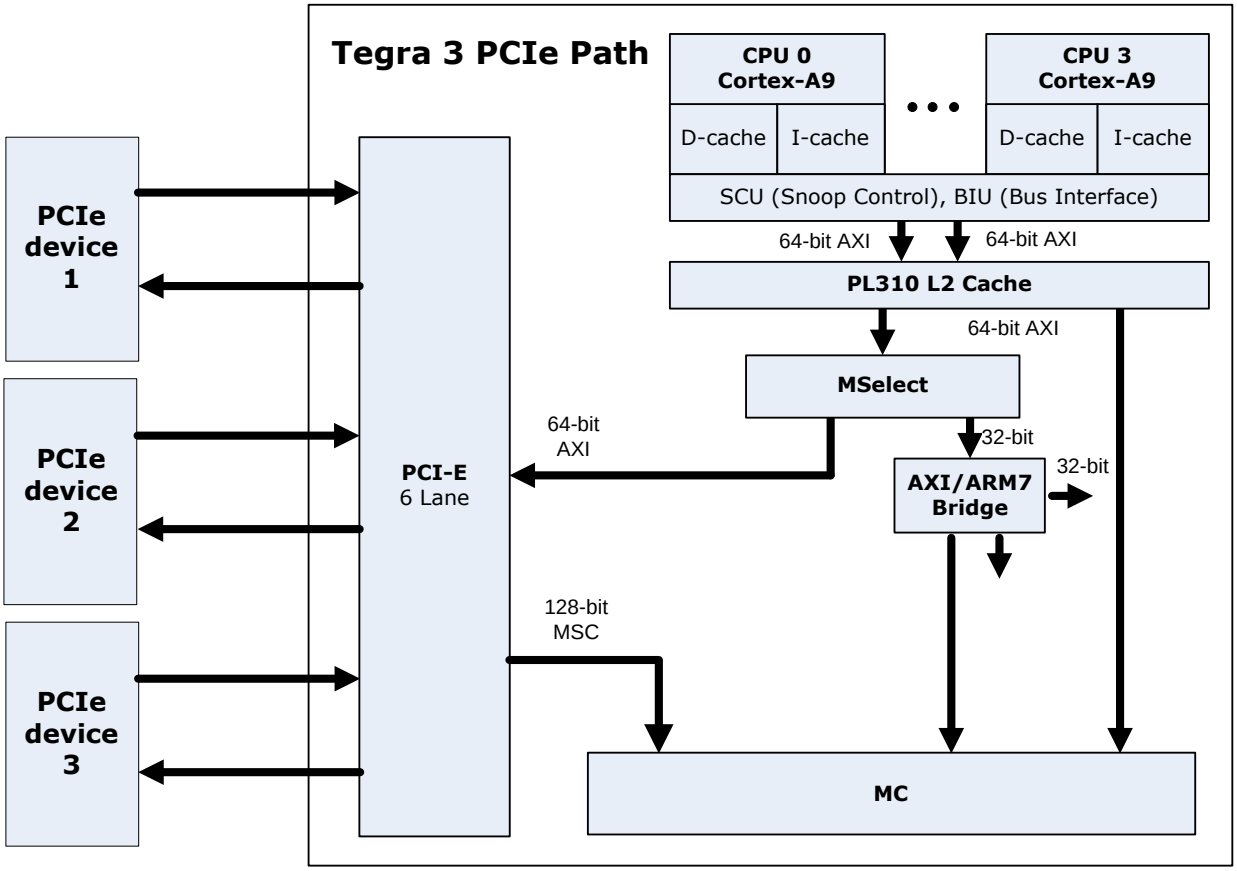
Bit	R/W	Reset	Description
31	RW	SKIP	PLLE_SS_SEQ_INCLUDE: 0=skip, 1=include. PLLE spread spectrum power sequencer include. 0 = SKIP 1 = INCLUDE
27:26	RO	X	PLLE_SEQ_STATE: PLLE power sequencer state 0 = SEQ_OFF 1 = SEQ_ON 2 = SEQ_BUSY
25	RW	SEQ_ON	PLLE_SEQ_START_STATE: PLLE power sequencer start state 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 0-255ms in 1ms steps.
15:8	RW	0x40	PLLE_LOCK_DLY: PT5: Delay from PLLE ENABLE to lock. Range 0-128us in 1us 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 us or ms, fast is per osc clock. Applies to all programmable timer counters in SATA, PCIe and PLLE seq.
6	RW	0x1	PLLE_SS_SWCTL: 0=PLLE spread spectrum config setup by HW, 1=setup by SW during training
5	RW	0x1	PLLE_CONFIG_SWCTL: 0=PLLE config setup by HW, 1=PLLE setup by SW during training. Affects resetable bits when PLL is off. SETUP and EXT_SETUP bits. All other config bit are left untouched as initialized. ex. M/N/P, SS coeff.
4	RW	0x1	PLLE_ENABLE_SWCTL: 0=PLLE enable by HW, 1=PLLE enable by SW
3	RW	0x0	PLLE_USE_LOCKDET: 0=Use programmable delay, 1=use lockdet signals from PLLE

Bit	R/W	Reset	Description
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 12MHz reference) 0 = OSC_DIV 1 = PLLP_OUT0
1	RW	DISABLE	PLLE_CML1_OEN: PLLE CML1 clock out enable. Used for SATA. 0 = DISABLE 1 = ENABLE

32.0 PCI EXPRESS (PCIe) ARCHITECTURE

The Tegra® 3 devices incorporate a 6-lane PCI Express® (PCIe®) bridge with support for both upstream and downstream FPCI interfaces that serves as the control path from Tegra 3 device to the external PCIe device. The PCIe bridge enables a connection to one, two, or three PCIe endpoints.

Figure 82 PCIe Bridge



32.1 Supported Configurations

Tegra 3 devices support the following lane/controller (number of lanes by number of controllers) configurations:

- 4×1, 1×2
- 4×1, 2×1
- 2×3

32.1.1 Device Recommendations

NVIDIA recommends that all PCIe devices used with Tegra 3 devices should support message signaled interrupt (MSI) signaling, and should be used in that mode. Correct operation of interrupts is ensured in this mode.

32.2 Registers

The PCI Express registers are located as follows:

Name	Location	Description
NV_PCIE_AXI_RP_T0C0	NV_ADDRESS_MAP_PCIE_BASE	Start of PCIe extended config space for controller 0
NV_PCIE_AXI_RP_T0C1	(NV_PCIE_AXI_RP_T0C0) + 4096	Start of PCIe extended config space for controller 1
NV_PCIE_AXI_PCA_UFPCI_T0C0	(NV_PCIE_AXI_RP_T0C1) + 4096	Start of PCIe pca0 space
NV_PCIE_AXI_PCA_XCLK_T0C0	(NV_PCIE_AXI_PCA_UFPCI_T0C0) + 256)	Start of PCIe pca1 space
NV_PCIE_AXI_PADS	(NV_PCIE_AXI_RP_T0C1) + 8192	Start of PCIe pads space
NV_PCIE_AXI_RP_T0C2	(NV_PCIE_AXI_PADS) + 4096	Start of PCIe extended config space for controller 2

32.2.1 PCI Compatible Configuration Registers

32.2.1.1 T_PCIE2_RP_DEV_ID (x000h)

This register implements the Device ID and Vendor ID registers as defined by the PCI™ 2.0 Specification.

Device ID and Vendor ID Register

Offset: 000h | Read/Write: RO

Bit	Reset	Description
31:16	None	**T_PCIE2_RP_DEV_ID_DEVICE_ID**: The DEVICE_ID bits identify the particular device. This identifier is allocated by the vendor. NVIDIA uses unique Device IDs for Root Complexes with different Maximum Link Widths. E1Ch **DEVICE_ID_TMS0_CTLR0_W4** E1Dh **DEVICE_ID_TMS0_CTLR1_W2 |** E1Dh **DEVICE_ID_TMS0_CTLR2_W2**
15:0	10DEh	**T_PCIE2_RP_DEV_ID_VENDOR_ID**: The VENDOR_ID bits identify the manufacturer of the device. Valid vendor identifiers are allocated by the PCI SIG to ensure uniqueness. 10DEh **VENDOR_ID_NVIDIA** (default)

32.2.1.2 T_PCIE2_RP_DEV_CTRL (x0004h)

Command and Status Register

Offset: 004h | Read/Write: R/W

Bit	R/W	Reset	Description
31	RW1C	0h	**T_PCIE2_RP_DEV_CTRL_DETECTED_PERR**: This bit is set by the Primary side device whenever it receives a Poisoned TLP, regardless of the state the Parity Error Response bit in the Command register. 0h **DETECTED_PERR_NOT_ACTIVE** (default) 1h **DETECTED_PERR_ACTIVE** 1h **DETECTED_PERR_SET**

Bit	R/W	Reset	Description
30	RW1C	0h	**T_PCIE2_RP_DEV_CTRL_SIGNALED_SERR**: This bit is set when the Primary side device sends an ERR_FATAL or ERR_NONFATAL message, and the SERR Enable bit is set. 0h **SIGNALED_SERR_NOT_ACTIVE** (default) 1h **SIGNALED_SERR_ACTIVE** 1h **SIGNALED_SERR_SET**
29	RW1C	0h	**T_PCIE2_RP_DEV_CTRL_RECEIVED_MASTER**: This bit is set when the Primary side Requestor receives a Completion with Unsupported Request Completion Status. 0h **RECEIVED_MASTER_NO_ABORT** (default) 1h **RECEIVED_MASTER_ABORT** 1h **RECEIVED_MASTER_SET**
28	RW1C	0h	**T_PCIE2_RP_DEV_CTRL_RECEIVED_TARGET**: This bit is set when the Primary side Requestor receives a Completion with Completer Abort Completion Status. 0h **RECEIVED_TARGET_NO_ABORT** (default) 1h **RECEIVED_TARGET_ABORT** 1h **RECEIVED_TARGET_SET**
27	RW1C	0h	**T_PCIE2_RP_DEV_CTRL_SIGNALED_TARGET**: This bit is set when the Primary side device completes a Request using Completer Abort Completion Status. 0h **SIGNALED_TARGET_NO_ABORT** (default) 1h **SIGNALED_TARGET_ABORT** 1h **SIGNALED_TARGET_SET**
26:25	R	0h	**T_PCIE2_RP_DEV_CTRL_DEVSEL_TIMING**: Does not apply to PCI Express. Hardwired to 0. 0h **DEVSEL_TIMING_FAST** (default) 1h **DEVSEL_TIMING_MEDIUM** 2h **DEVSEL_TIMING_SLOW**
24	RW1C	0h	**T_PCIE2_RP_DEV_CTRL_MASTER_DATA_PERR**: Master Data Parity Error bit. This bit is set by the Primary side Requestor if the Parity Error Response bit in the Command register is 1b and either of the following two conditions occurs: * Requestor receives a Completion marked poisoned * Requestor poisons a write Request If the Parity Error Response bit is 0b, this bit is never set. 0h **MASTER_DATA_PERR_NOT_ACTIVE** (default) 1h **MASTER_DATA_PERR_ACTIVE** 1h **MASTER_DATA_PERR_SET**
23	R	0h	**T_PCIE2_RP_DEV_CTRL_FAST_BACK2BACK**: Does not apply to PCI Express. Hardwired to 0. 0h **FAST_BACK2BACK_INCAPABLE** (default) 1h **FAST_BACK2BACK_CAPABLE**
22	R	0	Reserved
21	R	0h	**T_PCIE2_RP_DEV_CTRL_66MHZ**: Does not apply to PCI Express. Hardwired to 0. 0h **66MHZ_INCAPABLE** (default) 1h **66MHZ_CAPABLE**
20	R	1h	**T_PCIE2_RP_DEV_CTRL_CAPLIST**: The CAPLIST bit indicates that the device configuration space includes a capabilities list. Hardwired to 1. 1h **CAPLIST_PRESENT** (default) 0h **CAPLIST_NOT_PRESENT**
fs19	R	0h	**T_PCIE2_RP_DEV_CTRL_INTR_STATUS**: The INTR_STATUS bit indicates that an INTx interrupt message is pending internally to the device. 0h **INTR_STATUS_NOT_ACTIVE** (default) 1h **INTR_STATUS_ACTIVE**

Bit	R/W	Reset	Description
18:11	R	0	Reserved
10	R/W	0h	**T_PCIE2_RP_DEV_CTRL_INTR_DISABLE**: The INTR-DISABLE bit controls the ability of the device to generate INTx interrupt messages. When set, devices are prevented from generating INTx interrupt messages. 0h **INTR_DISABLE_INIT** (default) 1h **INTR_DISABLE_YES** 0h **INTR_DISABLE_NO**
9	R	0h	**T_PCIE2_RP_DEV_CTRL_BACK2BACK**: Does not apply to PCI Express. Hardwired to 0. 0h **BACK2BACK_DISABLED** (default) 1h **BACK2BACK_ENABLED**
8	R/W	0h	**T_PCIE2_RP_DEV_CTRL_SERR**: SERR Enable bit. This bit, when set, enables reporting of Non-fatal and Fatal errors detected by the Root Complex. In addition, this bit, when set, enables transmission by the primary interface of ERR_NONFATAL and ERR_FATAL error messages forwarded from the secondary interface. This bit does not affect the transmission of forwarded ERR_COR messages. 0h **SERR_DISABLED** (default) 1h **SERR_ENABLED**
7	R	0h	**T_PCIE2_RP_DEV_CTRL_STEP**: Does not apply to PCI Express. Hardwired to 0. 0h **STEP_DISABLED** (default) 1h **STEP_ENABLED**
6	R/W	0h	**T_PCIE2_RP_DEV_CTRL_PERR**: Parity Error Response bit. This bit, when set, controls the reporting of parity errors detected by the root complex (see Master Data Parity Error bit in the Status register). 0h **PERR_DISABLED** (default) 1h **PERR_ENABLED**
5	R	0h	**T_PCIE2_RP_DEV_CTRL_PALETTE_SNOOP**: Does not apply to PCI Express. Hardwired to 0. 0h **PALETTE_SNOOP_DISABLED** (default) 1h **PALETTE_SNOOP_ENABLED**
4	R	0h	**T_PCIE2_RP_DEV_CTRL_WRITE_AND_INVAL**: Does not apply to PCI Express. Hardwired to 0. 0h **WRITE_AND_INVAL_DISABLED** (default) 1h **WRITE_AND_INVAL_ENABLED**
3	R	0h	**T_PCIE2_RP_DEV_CTRL_SPECIAL_CYCLE**: Does not apply to PCI Express. Hardwired to 0. 0h **SPECIAL_CYCLE_DISABLED** (default) 1h **SPECIAL_CYCLE_ENABLED**
2	R/W	0h	**T_PCIE2_RP_DEV_CTRL_BUS_MASTER**: The BUS_MASTER bit controls the ability of the root complex to issue memory and I/O read/write requests. It controls forwarding of memory or I/O requests from the secondary interface to the primary interface. 0h **BUS_MASTER_DISABLED** (default) 1h **BUS_MASTER_ENABLED**
1	R/W	0h	**T_PCIE2_RP_DEV_CTRL_MEMORY_SPACE**: The MEMORY_SPACE bit indicates that the device will respond to memory space accesses on the primary interface. A value of 0 disables the device response on the primary interface. A value of 1 allows the device to forward memory transactions from the primary interface to the secondary interface. 0h **MEMORY_SPACE_DISABLED** (default) 1h **MEMORY_SPACE_ENABLED**

Bit	R/W	Reset	Description
0	R/W	0h	**T_PCIE2_RP_DEV_CTRL_IO_SPACE**: The IO_SPACE bit indicates that the device will respond to I/O space accesses on the primary interface. A value of 0 disables the device response on the primary interface. A value of 1 allows forwarding of I/O accesses from the primary interface to the secondary interface. 0h **IO_SPACE_DISABLED** (default) 1h **IO_SPACE_ENABLED**

32.2.1.3 T_PCIE2_RP_REV_CC (x0008h)

This register implements the Revision ID and Class Code registers as defined by the PCI 2.0 Specification.

Revision ID and Class Code Registers

Offset: 008h | Read/Write: RO

Bit	Reset	Description
31:8	6040XXh	**T_PCIE2_RP_REV_CC_CLASS_CODE**: 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 (bits 31:24) is a base class code which broadly classifies the type of function the device performs. The middle-byte (bits 23:16) is a sub-class code which identifies more specifically the function of the device. The lower byte (bits 15:8) identifies a specific register-level programming interface, if any, so that device independent software can interact with the device. 60400h **CLASS_CODE_HOST** (default) 60400h **CLASS_CODE_P2P**
7:0	None	**T_PCIE2_RP_REV_CC_REVISION_ID**: The REVISION_ID bits specify a device specific revision identifier. The top nibble is the Major Revision and the bottom nibble is the Minor Revision. The Major Revision is changed every time there is an all layer change to the Controller (A, B, C ...). The Minor Revision is updated through metal edits each time there is a metal revision changing the functionality of this block. A1h **REVISION_ID_VAL**

32.2.1.4 T_PCIE2_RP_MISC_1 (x000Ch)

This register implements the Cache Line Size, Latency Timer, Header Type, and BIST registers as defined by the PCI 2.0 Specification.

Cache Line Size, Latency Timer, Header Type, and BIST Registers

Offset: 00ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0h	**T_PCIE2_RP_MISC_1_BIST**: The BIST register field is optional and not used. Hardwired to 0. 0h **BIST_ZERO** (default)
23	R	0h	**T_PCIE2_RP_MISC_1_HEADER_TYPE1**: 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 23 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. 0h **HEADER_TYPE1_SINGLEFUNC** (default) 1h **HEADER_TYPE1_MULTIFUNC**

Bit	R/W	Reset	Description
22:16	R	1h	**T_PCIE2_RP_MISC_1_HEADER_TYPE0**: 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. Bits 22:16 in this register specify the layout of bytes 10h through 3Fh. PCI-Express Root Port Devices always employ Type 1 headers, hence this register is hard-wired to 1h. 0h **HEADER_TYPE0_NON_BRIDGE** 1h **HEADER_TYPE0_P2P_BRIDGE** (default)
15:11	R	0h	**T_PCIE2_RP_MISC_1_PLATENCY_TIMER**: The primary/master latency timer does not apply to PCI Express. Hardwired to 0. 0h **PLATENCY_TIMER_0_CLOCKS** (default)
10:8	R	0	Reserved
7:0	R/W	0h	**T_PCIE2_RP_MISC_1_CACHE_LINE_SIZE**: The cache line size register is set by the system firmware and the operating system to system cache line size. However, note that legacy PCI software may not always be able to program this field correctly especially in case of Hot-Plug devices. This field is implemented by PCI Express devices as a read-write field for legacy compatibility purposes but has no impact on any PCI Express device functionality. 0h **CACHE_LINE_SIZE_0_BYTES** (default)

32.2.1.5 T_PCIE2_RP_BAR_0 (x0010h)

This register implements the Base Address Register 0 as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2. It is unused and hard-wired to 0.

Base Address Register 0

Offset: 010h | Read/Write: RO

Bit	Reset	Description
31:0	0h	**T_PCIE2_RP_BAR_0_RESERVED**: This register is unused and hard-wired to 0. 0h **RESERVED_0** (default)

32.2.1.6 T_PCIE2_RP_BAR_1 (x0014h)

This register implements the Base Address Register 1 as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2. It is unused and hard-wired to 0.

Base Address Register 1

Offset: 014h | Read/Write: RO

Bit	Reset	Description
31:0	0h	**T_PCIE2_RP_BAR_1_RESERVED**: This register is unused and hard-wired to 0. 0h **RESERVED_0** (default)

32.2.1.7 T_PCIE2_RP_BN_LT (x0018h)

This register implements the Primary Bus Number, Secondary Bus Number, Subordinate Bus Number, and Secondary Latency Timer registers as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Primary Bus, Secondary Bus, Subordinate Bus Numbers, and Secondary Latency Timer Registers

Offset: 018h | Read/Write: R/W

Bit	R/W	Reset	Description
31:27	R	0h	**T_PCIE2_RP_BN_LT_SLATENCY_TIMER**: This register does not apply to PCI Express. Hardwired to 0. 0h **SLATENCY_TIMER_0_CLOCKS** (default)
26:24	R	0	Reserved
23:16	R/W	0h	**T_PCIE2_RP_BN_LT_SUB_BUS_NUMBER**: Subordinate Bus Number register is used to record the bus number of the highest numbered PCI bus segment which is behind the bridge Configuration software programs the value in this register. The bridge uses this register in conjunction with the Secondary Bus Number register to determine when to respond to a Type 1 configuration transaction on the primary interface. 0h **SUB_BUS_NUMBER_0** (default) 1h **SUB_BUS_NUMBER_1** 2h **SUB_BUS_NUMBER_2** FFh **SUB_BUS_NUMBER_255**
15:8	R/W	0h	**T_PCIE2_RP_BN_LT_SEC_BUS_NUMBER**: The Secondary Bus Number register is used to record the bus number of the PCI bus segment to which the secondary interface of the bridge is connected. Configuration software programs the value in this register. The bridge uses this register to decode Type 1 configuration transactions on the primary interface and convert them to Type 0 accesses on the secondary interface. 0h **SEC_BUS_NUMBER_0** (default) 1h **SEC_BUS_NUMBER_1** 2h **SEC_BUS_NUMBER_2** FFh **SEC_BUS_NUMBER_255**
7:0	R/W	0h	**T_PCIE2_RP_BN_LT_PRI_BUS_NUMBER**: The Primary Bus Number register is used to record the bus number of the PCI bus segment to which the primary interface of the bridge is connected. Configuration software programs the value in this register. The bridge uses this register to decode Type 1 configuration transactions on the secondary interface that must be converted to Special Cycle transactions on the primary interface. 0h **PRI_BUS_NUMBER_0** (default)

32.2.1.8 T_PCIE2_RP_IO_BL_SS (x001Ch)

This register implements the I/O Base, I/O Limit, and Secondary Status registers as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

The I/O Base and I/O Limit registers define an address range that is used by the bridge to determine when to forward I/O transactions from one interface to the other.

The Secondary Status register is similar in function and bit definition to the Status register defined above; however, its bits reflect status conditions of the secondary interface (the Status register reflects the status conditions of the primary interface)

I/O Base, I/O Limit and Secondary Status Registers

Offset: 01ch | Read/Write: R/W

Bit	R/W	Reset	Description
31	RW1C	0h	**T_PCIE2_RP_IO_BL_SS_DETECTED_PERR**: This bit is set when the Secondary side device receives a Poisoned TLP, regardless of the state the Parity Error Response bit in the Bridge Control register. 0h **DETECTED_PERR_NOT_ACTIVE** (default) 1h **DETECTED_PERR_ACTIVE** 1h **DETECTED_PERR_SET**

Bit	R/W	Reset	Description
30	RW1C	0h	**T_PCIE2_RP_IO_BL_SS_RECEIVED_SERR**: This bit is set when the Secondary side device receives an ERR_FATAL or ERR_NONFATAL message. 0h **RECEIVED_SERR_NOT_ACTIVE** (default) 1h **RECEIVED_SERR_ACTIVE** 1h **RECEIVED_SERR_SET**
29	RW1C	0h	**T_PCIE2_RP_IO_BL_SS_RECEIVED_MASTER**: This bit is set when the Secondary side device receives a Completion with Unsupported Request Completion Status. 0h **RECEIVED_MASTER_NO_ABORT** (default) 1h **RECEIVED_MASTER_ABORT** 1h **RECEIVED_MASTER_SET**
28	RW1C	0h	**T_PCIE2_RP_IO_BL_SS_RECEIVED_TARGET**: This bit is set when the Secondary side device receives a Completion with Completer Abort Completion Status. 0h **RECEIVED_TARGET_NO_ABORT** (default) 1h **RECEIVED_TARGET_ABORT** 1h **RECEIVED_TARGET_SET**
27	RW1C	0h	**T_PCIE2_RP_IO_BL_SS_SIGNALED_TARGET**: This bit is set when the Secondary side device completes a Request using Completer Abort Completion Status. 0h **SIGNALED_TARGET_NO_ABORT** (default) 1h **SIGNALED_TARGET_ABORT** 1h **SIGNALED_TARGET_SET**
26:25	R	0h	**T_PCIE2_RP_IO_BL_SS_DEVSEL_TIMING**: Does not apply to PCI Express. Hardwired to 0. 0h **DEVSEL_TIMING_FAST** (default) 1h **DEVSEL_TIMING_MEDIUM** 2h **DEVSEL_TIMING_SLOW**
24	RW1C	0h	**T_PCIE2_RP_IO_BL_SS_MASTER_DATA_PERR**: Master Data Parity Error bit. This bit is set by the Secondary side Requestor if the Parity Error Response Enable bit in the Bridge Control register is 1b and either of the following two conditions occurs: * Requestor receives a Completion marked poisoned * Requestor poisons a write Request If the Parity Error Response bit is 0b, this bit is never set. 0h **MASTER_DATA_PERR_NOT_ACTIVE** (default) 1h **MASTER_DATA_PERR_ACTIVE** 1h **MASTER_DATA_PERR_SET**
23	R	0h	**T_PCIE2_RP_IO_BL_SS_FAST_BACK2BACK**: Does not apply to PCI Express. Hardwired to 0. 0h **FAST_BACK2BACK_INCAPABLE** (default) 1h **FAST_BACK2BACK_CAPABLE**
22	R	0	Reserved
21	R	0h	**T_PCIE2_RP_IO_BL_SS_66MHZ**: Does not apply to PCI Express. Hardwired to 0. 0h **66MHZ_INCAPABLE** (default) 1h **66MHZ_CAPABLE**
20:16	R	0	Reserved

Bit	R/W	Reset	Description
15:12	R/W	0h	**T_PCIE2_RP_IO_BL_SS_IO_LIMIT**: This register corresponds to address bits AD[15:12] of the I/O limit. For the purpose of address decoding, the bridge assumes that the lower 12 address bits of the I/O limit are 0xFFF. The I/O Limit register can be programmed to a smaller value than the I/O Base register if there are no I/O addresses on the secondary side of the bridge. 0h **IO_LIMIT_ADDRESS_0** (default) 1h **IO_LIMIT_ADDRESS_256** 2h **IO_LIMIT_ADDRESS_512** Fh **IO_LIMIT_ADDRESS_64K**
11:8	R	1h	**T_PCIE2_RP_IO_BL_SS_IO_LIMIT_SUPPORT**: Identifies the I/O addressing support. When 0h, the device supports only 16-bit addressing. When 1h, it supports 32-bit addressing, in which case the I/O Limit Upper 16 Bits register is used to extend the I/O limit to 32 bits. Hard-wired to 1h. 0h **IO_LIMIT_SUPPORT_16** 1h **IO_LIMIT_SUPPORT_32** (default)
7:4	R/W	0h	**T_PCIE2_RP_IO_BL_SS_IO_BASE**: This register corresponds to address bits AD[15:12] of the I/O base address. For the purpose of address decoding, the bridge assumes that the lower 12 address bits, AD[11:0], of the I/O base address are zero. 0h **IO_BASE_ADDRESS_0** (default) 1h **IO_BASE_ADDRESS_256** 2h **IO_BASE_ADDRESS_512** Fh **IO_BASE_ADDRESS_64K**
3:0	R	1h	**T_PCIE2_RP_IO_BL_SS_IO_BASE_SUPPORT**: Identifies the I/O addressing support. When 0h, the device only supports 16-bit addressing. When 1h, it supports 32-bit addressing, in which case the I/O Base Upper 16 Bits register is used to extend the I/O base address to 32 bits. Hard-wired to 1h. 0h **IO_BASE_SUPPORT_16** 1h **IO_BASE_SUPPORT_32** (default)

32.2.1.9 T_PCIE2_RP_MEM_BL (x0020h)

This register implements the Memory Base and Memory Limit registers as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

The Memory Base and Memory Limit registers define an address range that is used by the bridge to determine when to forward memory-mapped I/O transactions from one interface to the other.

Memory Base and Memory Limit Registers

Offset: 020h | Read/Write: R/W

Bit	R/W	Reset	Description
31:20	R/W	0h	**T_PCIE2_RP_MEM_BL_MEM_LIMIT**: This register corresponds to address bits AD[31:20] of the memory-mapped I/O limit. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, AD[19:0], of the memory-mapped I/O limit are 0xFFFFF. The Memory Limit register must be programmed to a smaller value than the Memory Base register if there are no memory-mapped I/O addresses on the secondary side of the bridge. 0h **MEM_LIMIT_ADDRESS_0** (default) 1h **MEM_LIMIT_ADDRESS_1MEG** 2h **MEM_LIMIT_ADDRESS_2MEG** FFFh **MEM_LIMIT_ADDRESS_4GIG**
19:16	R	0	Reserved

Bit	R/W	Reset	Description
15:4	R/W	1h	**T_PCIE2_RP_MEM_BL_MEM_BASE**: This register corresponds to address bits AD[31:20] of the memory-mapped I/O base address. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, AD[19:0], of the memory-mapped I/O base address are zero. 0h **MEM_BASE_ADDRESS_0** 1h **MEM_BASE_ADDRESS_1MEG** (default) 2h **MEM_BASE_ADDRESS_2MEG** FFFh **MEM_BASE_ADDRESS_4GIG**
3:0	R	0	Reserved

32.2.1.10 T_PCIE2_RP_PRE_BL (x0024h)

This register implements the Prefetchable Memory Base and Prefetchable Memory Limit registers as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

The Prefetchable Memory Base and Prefetchable Memory Limit registers define an address range that is used by the bridge to determine when to forward prefetchable memory transactions from one interface to the other.

Prefetchable Memory Base and Prefetchable Memory Limit Registers

Offset: 024h | Read/Write: R/W

Bit	R/W	Reset	Description
31:20	R/W	0h	**T_PCIE2_RP_PRE_BL_PREFETCH_MEM_LIMIT**: This register corresponds to address bits AD[31:20] of the prefetchable memory limit. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, AD[19:0], of the prefetchable memory limit are 0xFFFFF. The Prefetchable Memory Limit register must be programmed to a smaller value than the Prefetchable Memory Base register if there is no prefetchable memory on the secondary side of the bridge. 0h **PREFETCH_MEM_LIMIT_ADDRESS_0** (default) 1h **PREFETCH_MEM_LIMIT_ADDRESS_1MEG** 2h **PREFETCH_MEM_LIMIT_ADDRESS_2MEG** FFFh **PREFETCH_MEM_LIMIT_ADDRESS_4GIG**
19:16	R	1h	**T_PCIE2_RP_PRE_BL_L64BIT**: Identifies the prefetchable memory addressing support. When 0h, the device supports only 32-bit addressing. When 1h, it supports 64-bit addressing, in which case the Prefetchable Limit Upper 32 Bits register is used to extend the prefetchable memory limit to 64 bits. Hard-wired to 1h. 1h **L64BIT_YES** (default)
15:4	R/W	1h	**T_PCIE2_RP_PRE_BL_PREFETCH_MEM_BASE**: This register corresponds to address bits AD[31:20] of the prefetchable memory base address. For the purpose of address decoding, the bridge assumes that the lower 20 address bits, AD[19:0], of the Prefetchable memory base address are zero. 0h **PREFETCH_MEM_BASE_ADDRESS_0** 1h **PREFETCH_MEM_BASE_ADDRESS_1MEG** (default) 2h **PREFETCH_MEM_BASE_ADDRESS_2MEG** FFFh **PREFETCH_MEM_BASE_ADDRESS_4GIG**
3:0	R	1h	**T_PCIE2_RP_PRE_BL_B64BIT**: Identifies the prefetchable memory addressing support. When 0h, the device supports only 32-bit addressing. When 1h, it supports 64-bit addressing, in which case the Prefetchable Base Upper 32 Bits register is used to extend the prefetchable memory base address to 64 bits. Hard-wired to 1h. 1h **B64BIT_YES** (default)

32.2.1.11 T_PCIE2_RP_PRE_BU32 (x0028h)

This register implements the Prefetchable Memory Base Upper 32 Bits register as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Prefetchable Memory Base Upper 32 Bits Register

Offset: 028h | Read/Write: R/W

Bit	Reset	Description
31:0	0h	**T_PCIE2_RP_PRE_BU32_BASE_UPPER_BITS**: This register specifies the upper 32 bits, corresponding to AD[63::32], of the 64-bit prefetchable memory base address. 0h **BASE_UPPER_BITS_0** (default)

32.2.1.12 T_PCIE2_RP_PRE_LU32 (x002Ch)

This register implements the Prefetchable Memory Limit Upper 32 Bits register as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Prefetchable Memory Limit Upper 32 Bits Register

Offset: 02ch | Read/Write: R/W

Bit	Reset	Description
31:0	0h	**T_PCIE2_RP_PRE_LU32_LIMIT_UPPER_BITS**: This register specifies the upper 32 bits, corresponding to AD[63::32], of the 64-bit prefetchable memory limit. 0h **LIMIT_UPPER_BITS_0** (default)

32.2.1.13 T_PCIE2_RP_IO_BL_U16 (x0030h)

This register implements the I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits registers as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

I/O Base Upper 16 Bits and I/O Limit Upper 16 Bits Registers

Offset: 030h | Read/Write: R/W

Bit	Reset	Description
31:16	0h	**T_PCIE2_RP_IO_BL_U16_LIMIT_UPPER_BITS**: This register specifies the upper 16 bits, corresponding to AD[31::16], of the 32-bit I/O limit. 0h **LIMIT_UPPER_BITS_0** (default)
15:0	0h	**T_PCIE2_RP_IO_BL_U16_BASE_UPPER_BITS**: This register specifies the upper 16 bits, corresponding to AD[31::16], of the 32-bit I/O base address. 0h **BASE_UPPER_BITS_0** (default)

32.2.1.14 T_PCIE2_RP_CAP_PTR (x0034h)

This register implements the Capabilities Pointer register as defined by the PCI 2.0 Specification.

Capabilities Pointer

Offset: 034h | Read/Write: RO

Bit	Reset	Description
31:8	0	Reserved
7:0	40h	**T_PCIE2_RP_CAP_PTR_CAP_PTR**: The CAP_PTR bits indicate the offset into configuration space where the capabilities list begins. 40h **CAP_PTR_PM** (default)

32.2.1.15 T_PCIE2_RP_ROM_BA (x0038h)

This register implements the optional Expansion ROM Base Address register as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2. It is unused and hard-wired to 0.

Expansion ROM Base Address Register

Offset: 038h | Read/Write: RO

Bit	Reset	Description
31:0	0h	**T_PCIE2_RP_ROM_BA_RESERVED**: This register is unused and hard-wired to 0. 0h **RESERVED_0** (default)

32.2.1.16 T_PCIE2_RP_INTR_BCR (x003Ch)

This register implements the Interrupt Line, Interrupt Pin, and Bridge Control registers as defined by the PCI 2.0 Specification and PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Interrupt Line, Interrupt Pin, and Bridge Control Registers

Offset: 03ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27	R	0h	**T_PCIE2_RP_INTR_BCR_DIS_TIMER_SERR**: Does not apply to PCI Express. Hard-wired to 0. 0h **DIS_TIMER_SERR_DISABLED** (default) 1h **DIS_TIMER_SERR_ENABLED**
26	R	0h	**T_PCIE2_RP_INTR_BCR_DIS_TIMER_STATUS**: Does not apply to PCI Express. Hard-wired to 0. 0h **DIS_TIMER_STATUS_NOT_ACTIVE** (default) 1h **DIS_TIMER_STATUS_ACTIVE**
25	R	0h	**T_PCIE2_RP_INTR_BCR_SECONDARY_DIS_TIMER**: Does not apply to PCI Express. Hard-wired to 0. 0h **SECONDARY_DIS_TIMER_LONG** (default) 1h **SECONDARY_DIS_TIMER_SHORT**

Bit	R/W	Reset	Description
24	R	0h	**T_PCIE2_RP_INTR_BCR_PRIMARY_DIS_TIMER**: Does not apply to PCI Express. Hard-wired to 0. 0h **PRIMARY_DIS_TIMER_LONG** (default) 1h **PRIMARY_DIS_TIMER_SHORT**
23	R	0h	**T_PCIE2_RP_INTR_BCR_FAST_B2B**: Does not apply to PCI Express. Hard-wired to 0. 0h **FAST_B2B_DISABLED** (default) 1h **FAST_B2B_ENABLED**
22	R/W	0h	**T_PCIE2_RP_INTR_BCR_SB_RESET**: Setting this bit triggers a hot reset on the corresponding PCI Express Port. 0h **SB_RESET_DISABLED** (default) 1h **SB_RESET_ENABLED**
21	R	0h	**T_PCIE2_RP_INTR_BCR_MABORT**: Does not apply to PCI Express. Hard-wired to 0. 0h **MABORT_DISABLED** (default) 1h **MABORT_ENABLED**
20	R/W	0h	**T_PCIE2_RP_INTR_BCR_VGA_16BITIO**: When Enabled, allows full 16-bit decode of IO address range for VGA. 0h **VGA_16BITIO_DISABLED** (default) 1h **VGA_16BITIO_ENABLED**
19	R/W	0h	**T_PCIE2_RP_INTR_BCR_VGA_ADDRESS**: When enabled the P2P bridge will claim all of the legacy VGA addresses. Memory address range: 000A_0000 - 000B_FFFF IO address ranges: 3B0-3BB, 3C0-3DF (including all aliases if VGA_16BITIO is not set). 0h **VGA_ADDRESS_DISABLED** (default) 1h **VGA_ADDRESS_ENABLED**
18	R/W	0h	**T_PCIE2_RP_INTR_BCR_ISA_ADDRESS**: Modifies the response by the bridge to ISA I/O addresses. This applies only to I/O addresses that are enabled by the I/O Base and I/O Limit registers and are in the first 64 KB of PCI I/O address space (0000 0000h to 0000 FFFFh). If this bit is set, the bridge will block any forwarding from primary to secondary of I/O transactions addressing the last 768 bytes in each 1-KB block. 0h **ISA_ADDRESS_DISABLED** (default) 1h **ISA_ADDRESS_ENABLED**
17	R/W	0h	**T_PCIE2_RP_INTR_BCR_SERR_FORWARD**: This bit controls forwarding of ERR_COR, ERR_NONFATAL and ERR_FATAL from secondary to primary. 0h **SERR_FORWARD_DISABLED** (default) 1h **SERR_FORWARD_ENABLED**
16	R/W	0h	**T_PCIE2_RP_INTR_BCR_PERR_RESP**: This bit controls the response to Poisoned TLPs. 0h **PERR_RESP_DISABLED** (default) 1h **PERR_RESP_ENABLED**
15:8	R	1h	**T_PCIE2_RP_INTR_BCR_INTR_PIN**: This register contains 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. This register is read-only. 0h **INTR_PIN_NONE** 1h **INTR_PIN_INTA** (default) 2h **INTR_PIN_INTB** 3h **INTR_PIN_INTC** 4h **INTR_PIN_INTD**

Bit	R/W	Reset	Description
7:0	R/W	0h	**T_PCIE2_RP_INTR_BCR_INTR_LINE**: This register contains 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 written to 0xff (no connection) by software if the bridge does not use an interrupt pin. Some PCI BIOSes cannot handle aliased INTR_LINEs. Some PCI BIOSes cannot handle INTR_LINE initialized to 0xff. 0h **INTR_LINE_IRQ0** (default) 1h **INTR_LINE_IRQ1** Fh **INTR_LINE_IRQ15** FFh **INTR_LINE_UNKNOWN**

32.2.2 PCI Subsystem ID and Subsystem Vendor ID Capability Registers

32.2.2.1 T_PCIE2_RP_SS_0 (x0040h)

This register implements the first register of the Subsystem ID and Subsystem Vendor ID Capability list item, as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Subsystem ID and Subsystem Vendor ID Capability Register 0

Offset: 040h | Read/Write: RO

Bit	Reset	Description
31:16	0	Reserved
15:8	48h	**T_PCIE2_RP_SS_0_NEXT_PTR**: This read-only field points to the next capabilities list item. 48h **NEXT_PTR_PM** (default)
7:0	Dh	**T_PCIE2_RP_SS_0_CAP_ID**: This read-only field is used to detect the presence of the SSID/SSVID registers in a PCI-to-PCI bridge. Hard-wired to 0Dh. Dh **CAP_ID_SS** (default)

32.2.2.2 T_PCIE2_RP_SS_1 (x0044h)

This register implements the second register of the .Subsystem ID and Subsystem Vendor ID Capability list item, as defined by the PCI-to-PCI Bridge Architecture Specification, Revision 1.2.

Subsystem ID and Subsystem Vendor ID Capability Register 1

Offset: 044h | Read/Write: RO

Bit	Reset	Description
31:16	0h	**T_PCIE2_RP_SS_1_SSID**: The SSID identifies the particular add-in card or subsystem and is assigned by the vendor. 0h **SSID_INIT** (default)
15:0	10DEh	**T_PCIE2_RP_SS_1_SSVID:** The SSVID identifies the manufacturer of the add-in card or subsystem. The SSVID is assigned by PCI-SIG to insure uniqueness (the Vendor ID is used as the SSVID also). 10DEh SSVID_INIT (default)

32.2.3 PCI Power Management Capability Structure Registers

32.2.3.1 T_PCIE2_RP_PM_0 (x0048h)

This register implements the Power Management Capabilities register as defined in Section 7.6 of PCI Express Specifications.

Power Management Capabilities Register

Offset: 048h | Read/Write: RO

Bit	Reset	Description
31:27	1Fh	**T_PCIE2_RP_PM_0_PME_SUPPORT:** This 5-bit field indicates the power states in which the function may assert PME#. A value of 0b for any bit indicates that the function is not capable of asserting the PME# signal while in that power state. bit(11) XXXX1b = PME# can be asserted from D0 bit(12) XXX1Xb = PME# can be asserted from D1 bit(13) XX1XXb = PME# can be asserted from D2 bit(14) X1XXXb = PME# can be asserted from D3hot bit(15) 1XXXXb = PME# can be asserted from D3cold Bits 31, 30, and 27 must be set to 1b for PCI-PCI Bridge structures representing Ports on Root Complexes/Switches to indicate that the Bridge will forward PME Messages. 1Fh PME_SUPPORT_YES (default) 0h PME_SUPPORT_NO
26	0h	**T_PCIE2_RP_PM_0_D2_SUPPORT**: If this bit is a "1", this function supports the D2 Power Management State. 1h **D2_SUPPORT_YES** 0h **D2_SUPPORT_NO** (default)
25	0h	**T_PCIE2_RP_PM_0_D1_SUPPORT**: If this bit is a "1", this function supports the D1 Power Management State. 1h **D1_SUPPORT_YES** 0h **D1_SUPPORT_NO** (default)
24:22	0h	**T_PCIE2_RP_PM_0_AUX_CURRENT:** This 3 bit field reports the 3.3Vaux auxiliary current requirements for the PCI function. Bit 3.3Vaux 8 7 6 Max. Current Required 1 1 1 375 mA 1 1 0 320 mA 1 0 1 270 mA 1 0 0 220 mA 0 1 1 160 mA 0 1 0 100 mA 0 0 1 55 mA 0 0 0 0 (self powered) 0h AUX_CURRENT_0 (default)
21	0h	**T_PCIE2_RP_PM_0_DEV_SPEC_INIT**: The Device Specific Initialization bit indicates whether special initialization of this function is required (beyond the standard PCI configuration header) before the generic class device driver is able to use it. Note that this bit is not used by some operating systems. Microsoft Windows and Windows NT, for instance, do not use this bit to determine whether to use D3. Instead, they use the driver's capabilities to determine this. A "1" indicates that the function requires a device specific initialization sequence following transition to the D0 uninitialized state. 0h **DEV_SPEC_INIT_NOT_NEEDED** (default) 1h **DEV_SPEC_INIT_NEEDED**
20	0	Reserved

Bit	Reset	Description
19	0h	**T_PCIE2_RP_PM_0_PME_CLOCK**: Does not apply to PCI Express. Must be hardwired to 0. 0h **PME_CLOCK_NOT_NEEDED** (default) 1h **PME_CLOCK_NEEDED**
18:16	3h	**T_PCIE2_RP_PM_0_PCIPM_REV**: A value of 010b indicates that this function complies with Revision 1.1 of the PCI Power Management Interface Specification. 3h **PCIPM_REV_12** (default) 2h **PCIPM_REV_11**
15:8	50h	**T_PCIE2_RP_PM_0_NEXT_PTR**: This read-only field points to the next capabilities list item. 50h **NEXT_PTR_MSI** (default)
7:0	1h	**T_PCIE2_RP_PM_0_CAP_ID**: This read-only field is used to detect the presence of the Power Management Capability registers in a PCI-to-PCI bridge. Hard-wired to 01h. 1h **CAP_ID_PM** (default)

32.2.3.2 T_PCIE2_RP_PM_1 (x004Ch)

This register implements the Power Management Status/Control register as defined in Section 7.6 of PCI Express Specification.

Power Management Status/Control Register

Offset: 04ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0h	**T_PCIE2_RP_PM_1_PME_DATA**: The PME Data register is not implemented. Hard-wired to 0. 0h **PME_DATA_UNS** (default)
23	R	0h	**T_PCIE2_RP_PM_1_PME_BPCC:** The bus power/clock control mechanism is not used. Hard-wired to 0. 0h **PME_BPCC_UNS** (default)
22	R	0h	**T_PCIE2_RP_PM_1_PME_B2B3:** The bus power/clock control mechanism is not used. Therefore the B2/B3 support bit is hard-wired to 0. 0h **PME_B2B3_UNS** (default)
21:16	R	0	Reserved
15	RW1C	0h	**T_PCIE2_RP_PM_1_PME_STATUS**: This bit is set when the function would normally assert the PME# signal independent of the state of the PME_En bit. Writing a "1" to this bit will clear it and cause the function to stop asserting a PME# (if enabled). Writing a "0" has no effect. This bit is sticky and must be explicitly cleared by the operating system each time the operating system is initially loaded. NOTE: This field is reset by Cold Reset 0h **PME_STATUS_NOT_ACTIVE** (default) 1h **PME_STATUS_ACTIVE** 1h **PME_STATUS_SET**
14:13	R	0h	**T_PCIE2_RP_PM_1_PME_DATA_SCALE:** The PME Data register is not implemented. Hard-wired to 0. 0h **PME_DATA_SCALE_UNS** (default)

Bit	R/W	Reset	Description
12:9	R	0h	**T_PCIE2_RP_PM_1_PME_DATA_SEL**: The PME Data register is not implemented. Hard-wired to 0. 0h **PME_DATA_SEL_UNS** (default)
8	R/W	0h	**T_PCIE2_RP_PM_1_PME:** A "1" enables the function to assert PME#. When "0", PME# assertion is disabled. This bit is sticky and must be explicitly cleared by the operating system each time it is initially loaded. NOTE: This field is reset by Cold Reset 0h PME_DISABLE (default) 1h PME_ENABLE
7:2	R	0	Reserved
1:0	R/W	0h	**T_PCIE2_RP_PM_1_PWR_STATE:** This 2-bit field is used both to determine the current power state of a function and to set the function into a new power state. The definition of the field values is given below. 00b - D0 01b - D1 10b - D2 11b - D3hot If software attempts to write an unsupported, optional state to this field, the write operation must complete normally on the bus; however, the data is discarded and no state change occurs. 0h **PWR_STATE_D0** (default) 1h **PWR_STATE_D1** 2h **PWR_STATE_D2** 3h **PWR_STATE_D3HOT**

32.2.4 PCI MSI Capability Structure Registers

32.2.4.1 T_PCIE2_RP_MSI_CTRL (x0050h)

The MSI capability structure is required for all PCI Express devices that are capable of generating interrupts.

MSI enables a device to request service by writing a system-specified message to a system-specified address. The transaction address specifies the message destination and the transaction data specifies the message. System software initializes the message destination and message during device configuration.

MSI Control and Capability Registers

Offset: 050h | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0h	**T_PCIE2_RP_MSI_CTRL_RSVD:** The RSVD field is reserved by the specification. 0h **RSVD_0** (default)
23	R	1h	**T_PCIE2_RP_MSI_CTRL_64BIT_CAP:** The 64BIT_CAP field indicates if the function is capable of generating a 64-bit message address. If 1, the function is capable. 1h **64BIT_CAP_TRUE** (default)
22:20	R/W	0h	**T_PCIE2_RP_MSI_CTRL_MULT_EN:** The MULT_EN field indicates the number of allocated messages. It is always aligned to a power of 2. 0h **MULT_EN_CODE0** (default) 1h **MULT_EN_CODE2** 2h **MULT_EN_CODE4** 3h **MULT_EN_CODE8**
19:17	R	1h	**T_PCIE2_RP_MSI_CTRL_MULT_CAP**: The MULT_CAP field determines the number of requested messages. It should always be a power of 2.

Bit	R/W	Reset	Description
			1h **MULT_CAP_CODE2** (default)
16	R/W	0h	**T_PCIE2_RP_MSI_CTRL_MSI:** The MSI bit controls if the function is permitted to use message signaled interrupt to request service. If 1, the function is permitted to use MSI and prohibits the use of INTx messages. 0h **MSI_DISABLE** (default) 1h **MSI_ENABLE**
15:8	R	None	**T_PCIE2_RP_MSI_CTRL_NEXT_PTR:** The NEXT_PTR field points to the next item in the capabilities list. 60h **NEXT_PTR_MSIMAP** 80h **NEXT_PTR_PCIEXP**
7:0	R	5h	**T_PCIE2_RP_MSI_CTRL_CAP_ID:** This read-only field is used to detect the presence of the Message Signaled Interrupt Capability registers in a PCI-to-PCI bridge. Hard-wired to 05h. 5h **CAP_ID_MSI** (default)

32.2.4.2 T_PCIE2_RP_MSI_LOW_ADDR (x0054h)

The contents of this register specify the lower 32 bits of the dword aligned address for the MSI memory write transaction.

MSI Message Lower Address Register

Offset: 054h | Read/Write: R/W

Bit	R/W	Reset	Description
31:2	R/W	0h	**T_PCIE2_RP_MSI_LOW_ADDR_DWORD**: Bits 31:2 of the address. 0h **DWORD_0** (default)
1:0	R	0h	**T_PCIE2_RP_MSI_LOW_ADDR_RSVD**: The address is always DWORD aligned, hence bits 1:0 of this register are hard-wired to 0. 0h **RSVD_0** (default)

32.2.4.3 T_PCIE2_RP_MSI_UPPER_ADDR (x0058h)

The contents of this register specify the upper 32 bits of the 64-bit MSI message address.

MSI Message Upper Address Register

Offset: 058h | Read/Write: R/W

Bit	Reset	Description
31:0	0h	**T_PCIE2_RP_MSI_UPPER_ADDR_DWORD**: Bits 63:32 of the address. 0h **DWORD_0** (default)

32.2.4.4 T_PCIE2_RP_MSI_DATA (x005Ch)

The contents of this register specify the data that is written to the MSI message address.

MSI Message Data Register

Offset: 05ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R	0h	**T_PCIE2_RP_MSI_DATA_RSVD**: 0h **RSVD_0** (default)
15:0	R/W	0h	**T_PCIE2_RP_MSI_DATA_DATA**: The MSI message data. 0h **DATA_0** (default)

32.2.5 PCI Express Capability Structure Registers

32.2.5.1 T_PCIE2_RP_PCI_EXPRESS_CAPABILITY (x0080h)

The PCI Express Capability List register enumerates the PCI Express Capability Structure in PCI 2.3 configuration space capability list. It also identifies PCI Express device type and associated capabilities.

PCI Express Capability List Register

Offset: 080h | Read/Write: RO

Bit	Reset	Description
31:30	0	Reserved
29:25	0h	**T_PCIE2_RP_PCI_EXPRESS_CAPABILITY_INTERRUPT_MESSAGE_NUMBER**: If this function is allocated more than one MSI interrupt number, this register is required to contain the offset between the base Message Data and the MSI Message that is generated when any of the status bits in either the Slot Status register or the Root Port Status register of this capability structure are set. Hardware is required to update this field so that it is correct if the number of the MSI Messages assigned to the device changes. 0h **INTERRUPT_MESSAGE_NUMBER_ZERO** (default)
24	1h	**T_PCIE2_RP_PCI_EXPRESS_CAPABILITY_SLOT_IMPLEMENTED**: This bit when set indicates that the PCI Express Link associated with this port is connected to a slot (as compared to being connected to an integrated component or being disabled). This field is valid for the following PCI Express device/Port Types: * Root Port of PCI Express Root Complex. * Downstream Port of PCI Express Switch. 1h **SLOT_IMPLEMENTED_INIT** (default)
23:20	4h	**T_PCIE2_RP_PCI_EXPRESS_CAPABILITY_DEVICE_PORT_TYPE:** Indicates the type of PCI Express logical device. Defined encodings are: 0000b = PCI Express Endpoint device. 0001b = Legacy PCI Express Endpoint device. 0100b = Root Port of PCI Express Root Complex. 0101b = Upstream Port of PCI Express Switch. 0110b = Downstream Port of PCI Express Switch. 0111b = PCI Express-to-PCI/PCI-X Bridge. 1000b = PCI/PCI-X to PCI Express Bridge **4h DEVICE_PORT_TYPE_INIT (default)**

Bit	Reset	Description
19:16	2h	**T_PCIE2_RP_PCI_EXPRESS_CAPABILITY_VERSION**: Indicates PCI_SIG defined PCI Express capability structure version number. Must be 1h for versions 1.0a and 1.1 of the PCI-Express Specification. Must be 2h for devices compliant to the Express Capabilities Register Expansion ECN. 2h **VERSION_INIT** (default) 1h **VERSION_1** 2h **VERSION_2**
15:8	0h	**T_PCIE2_RP_PCI_EXPRESS_CAPABILITY_LIST_NEXT_CAPABILITY_PTR**: The offset to the next PCI capability structure or 0h if no other items exist in the linked list of capabilities. 0h **LIST_NEXT_CAPABILITY_PTR_INIT** (default)
7:0	10h	**T_PCIE2_RP_PCI_EXPRESS_CAPABILITY_LIST_CAPABILITY_ID**: Indicates PCI Express Capability Structure. This field must return a Capability ID of 10h indicating that this is a PCI Express Capability Structure. 10h **LIST_CAPABILITY_ID_INIT** (default)

32.2.5.2 T_PCIE2_RP_DEVICE_CAPABILITY (x0084h)

The Device Capabilities register identifies PCI Express device specific capabilities.

Device Capabilities Register

Offset: 084h | Read/Write: RO

Bit	Reset	Description
31:28	0	Reserved
27:26	0h	**T_PCIE2_RP_DEVICE_CAPABILITY_CAPTURED_SLOT_POWER_LIMIT_SCALE**: Specifies the scale used for the Slot Power Limit Value. Range of Values: 00b = 1.0x 01b = 0.1x 11b = 0.01x 11b = 0.001x This value is set by the Set_Slot_Power_Limit message or hardwired to 00b. The default value is all 00b. 0h **CAPTURED_SLOT_POWER_LIMIT_SCALE_INIT** (default)
25:18	0h	**T_PCIE2_RP_DEVICE_CAPABILITY_CAPTURED_SLOT_POWER_LIMIT_VALUE**: In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by slot. Power limit (in Watts) is calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field. This value is set by the Set_Slot_Power_Limit message or hardwired to 0000_0000b. The default value is 0000_0000b. 0h **CAPTURED_SLOT_POWER_LIMIT_VALUE_INIT** (default)
17:16	0	Reserved
15	1h	**T_PCIE2_RP_DEVICE_CAPABILITY_ROLE_BASED_ERR_REPORTING**: This bit, when set, indicates that the device implements the functionality originally defined in the Error Reporting ECN for PCI Express Base Spec 1.0a and later incorporated into PCI Express Base Spec 1.1. 1h **ROLE_BASED_ERR_REPORTING_INIT** (default)
14	0h	**T_PCIE2_RP_DEVICE_CAPABILITY_POWER_INDICATOR_PRESENT**: This bit, when set, indicates that a Power Indicator is implemented on the card or module. 0h **POWER_INDICATOR_PRESENT_INIT** (default)
13	0h	**T_PCIE2_RP_DEVICE_CAPABILITY_ATTENTION_INDICATOR_PRESENT**: This bit, when set, indicates that an Attention Indicator is implemented on the card or module. 0h **ATTENTION_INDICATOR_PRESENT_INIT** (default)

Bit	Reset	Description
12	0h	**T_PCIE2_RP_DEVICE_CAPABILITY_ATTENTION_BUTTON_PRESENT**: This bit when set indicates that an Attention Button is implemented on the card or module. 0h **ATTENTION_BUTTON_PRESENT_INIT** (default)
11:9	0h	**T_PCIE2_RP_DEVICE_CAPABILITY_ENDPOINT_L1_ACCEPTABLE_LATENCY:** This field indicates acceptable latency that an Endpoint can withstand due to the transition from L1 state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering. • 000b -- Less than 1 us. • 001b -- 1 us to less than 2 us. • 010b -- 2 us to less than 4 us. • 011b -- 4 us to less than 8 us. • 100b -- 8 us to less than 16 us. • 101b -- 16 us to less than 32 us. • 110b -- 32 us-64 us. • 111b -- More than 64 us. 0h **ENDPOINT_L1_ACCEPTABLE_LATENCY_INIT** (default)
8:6	0h	**T_PCIE2_RP_DEVICE_CAPABILITY_ENDPOINT_L0S_ACCEPTABLE_LATENCY**: This field indicates the acceptable total latency that an Endpoint can withstand due to the transition from L0s state to the L0 state. It is essentially an indirect measure of the Endpoint's internal buffering. • 000b -- Less than 64 ns. • 001b -- 64 ns to less than 128 ns. • 010b -- 128 ns to less than 256 ns. • 011b -- 256 ns to less than 512 ns. • 100b -- 512 ns to less than 1 us. • 101b -- 1 us to less than 2 us. • 110b -- 2 us-4us. • 111b -- More than 4 us. 0h **ENDPOINT_L0S_ACCEPTABLE_LATENCY_INIT** (default)
5	None	**T_PCIE2_RP_DEVICE_CAPABILITY_EXTENDED_TAG_FIELD_SIZE**: This field indicates the maximum supported size of the Tag field. Defined encodings are: 0b = 5-bit Tag field supported. 1b = 8-bit Tag field supported.
4:3	0h	**T_PCIE2_RP_DEVICE_CAPABILITY_PHANTOM_FUNCTIONS_SUPPORTED:** This field indicates the support for use of unclaimed function numbers to extend the number of outstanding transactions allowed by logically combining unclaimed function numbers (called Phantom Functions) with the Tag identifier. This field indicates the number of most significant bits of the function number portion of Requester ID that are logically combined with the Tag identifier. Defined encodings are: • 00b -- No function number bits used for Phantom Functions; device may implement all function numbers. • 01b -- First most significant bit of the function number in Requester ID used for Phantom Functions; device may implement functions 0-3. Functions 0, 1, 2 and 3 may claim functions 4, 5, 6, and 7 as Phantom Functions respectively. • 10b -- First two most significant bits of the function number in Requestor ID used for Phantom Functions; device may implement functions 0-1. Function 0 may claim functions 2, 4 and 6 as Phantom Functions, function 1 may claim functions 3, 5 and 7 as Phantom Functions. • 11b -- All three bits of function number in Requester ID used for Phantom Functions; device must be a single function 0 device that may claim all other functions as Phantom Functions. **Note:** The Phantom Function support for the device must be enabled by the corresponding control field in the Device Control Register. 0h PHANTOM_FUNCTIONS_SUPPORTED_INIT (default)

Bit	Reset	Description
2:0	0h	**T_PCIE2_RP_DEVICE_CAPABILITY_MAX_PAYLOAD_SIZE:** This field indicates the maximum payload size that the device can support for TLP's. • 000b --128B max payload size • 001b -- 256B max payload size • 010b -- 512B max payload size • 011b -- 1024B max payload size • 100b -- 2048B max payload size • 101b -- 4096B max payload size • 110b -- Reserved • 111b -- Reserved 0h MAX_PAYLOAD_SIZE_INIT (default)

32.2.5.3 T_PCIE2_RP_DEVICE_CONTROL_STATUS (x0088h)

The Device Control register controls PCI Express device specific parameters. It also provides information about PCI Express device specific parameters.

Device Control and Device Status Registers

Offset: 088h | Read/Write: R/W

Bit	R/W	Reset	Description
31:22	R	0	Reserved
21	R	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_TRANSACTIONS_PENDING:** This bit when set indicates that a device has issued Non Posted requests which have not been completed. A device reports this bit cleared only when all Completions for any outstanding Non-Posted Requests have been received. 0h TRANSACTIONS_PENDING_INIT (default)
20	R	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_AUX_POWER_DETECTED:** Devices that require AUX power report this bit as set if AUX power is detected by the device. 0h **AUX_POWER_DETECTED_INIT** (default)
19	RW1C	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_UNSUPP_REQUEST_DETECTED:** This bit indicates that the device received an Unsupported Request. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control Register. NOTE: This field is reset by Cold Reset 0h **UNSUPP_REQUEST_DETECTED_INIT (default)** **1h UNSUPP_REQUEST_DETECTED_SET**
18	RW1C	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_FATAL_ERROR_DETECTED:** This bit indicates status of fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For devices supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the correctable error mask register. 0h **FATAL_ERROR_DETECTED_INIT** (default) 1h **FATAL_ERROR_DETECTED_SET**
17	RW1C	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_NON_FATAL_ERROR_DETECTED:** This bit indicates status of non-fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For devices supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the correctable error mask register. NOTE: This field is reset by Cold Reset 0h **NON_FATAL_ERROR_DETECTED_INIT** (default) 1h **NON_FATAL_ERROR_DETECTED_SET**

Bit	R/W	Reset	Description
16	RW1C	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_CORR_ERROR_DETECTED:** This bit indicates status of correctable errors detected. Errors are logged in this register regardless of whether error reporting is enabled or not in the Device Control register. For devices supporting Advanced Error Handling, errors are logged in this register regardless of the settings of the correctable error mask register. Default value of this field is 0. NOTE: This field is reset by Cold Reset 0h **CORR_ERROR_DETECTED_INIT** (default) 1h **CORR_ERROR_DETECTED_SET**
15	R	0	Reserved
14:12	R/W	2h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_MAX_READ_REQUEST_SIZE:** This field sets the maximum Read Request size for the Device as a Requester. The Device must not generate read requests with size exceeding the set value. Defined encodings for this field are: • 000b -- 128B max payload size • 001b -- 256B max payload size • 010b -- 512B max payload size • 011b -- 1024B max payload size • 100b -- 2048B max payload size • 101b -- 4096B max payload size • 110b -- Reserved • 111b – Reserved Devices that do not generate Read Requests larger than 128 bytes are permitted to implement this field as Read Only (RO) with a value of 000b. 2h **MAX_READ_REQUEST_SIZE_INIT** (default)
11	R/W	1h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_ENABLE_NO_SNOOP:** If this bit is set to 1, the device is permitted to set the No Snoop Bit in the Requester Attributes of transactions it initiates that do not require hardware enforced cache coherency. Even when this bit is set to 1, a device may only set the No Snoop attribute on a transaction when it can guarantee that the address of the transaction is not stored in any cache in the system. This bit may be hardwired to 0 if a device never sets the No Snoop attribute in transactions it initiates. 1h **ENABLE_NO_SNOOP_INIT (default)**
10	R/W	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_AUXILLARY_POWER_PM_ENABLE:** This bit when set enables a device to draw AUX power independent of PME AUX power. AUX power is allocated as requested in the AUX_Current field of the Power Management Capabilities Register (PMC), independent of the PME_En bit in the Power Management Control/Status Register (PMCSR). For multi-function devices, a component is allowed to draw AUX power if at least one of the functions has this bit set. Devices that do not implement this capability hardwire this bit to 0. NOTE: This field is reset by Cold Reset 0h **AUXILLARY_POWER_PM_ENABLE_INIT** (default)
9	R	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_PHANTOM_FUNCTIONS_ENABLE:** When set, this bit enables a device to use unclaimed functions as Phantom Functions to extend the number of outstanding transaction identifiers. If the bit is cleared, the device is not allowed to use Phantom Functions. Devices that do not implement this capability hardwire this bit to 0. 0h **PHANTOM_FUNCTIONS_ENABLE_INIT** (default)
8	R/W	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_EXTENDED_TAG_FIELD_ENABLE:** When set, this bit enables a device to use an 8-bit Tag field as a requester. If the bit is cleared, the device is restricted to a 5-bit Tag field. Devices that do not implement this capability hardwire this bit to 0. 0h **EXTENDED_TAG_FIELD_ENABLE_INIT** (default)

Bit	R/W	Reset	Description
7:5	R/W	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_MAX_PAYLOAD_SIZE:** This field sets the maximum TLP payload size for the device. As a receiver, the device must handle TLP's as large as the set value; as transmitter, the device must not generate TLPs exceeding the set value. Permissible values that can be programmed are indicated by the Max_Payload_Size supported in the Device Capabilities register. Defined encodings for this field are: • 000b -- 128B max payload size • 001b -- 256B max payload size • 010b -- 512B max payload size • 011b -- 1024B max payload size • 100b -- 2048B max payload size • 101b -- 4096B max payload size • 110b -- Reserved • 111b -- Reserved 0h **MAX_PAYLOAD_SIZE_INIT** (default)
4	R/W	1h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_ENABLE_RELAXED_ORDERING:** If this bit is set, the device is permitted to set the Relaxed Ordering bit in the Attributes field of transactions it initiates that do not require strong write ordering. This bit may be hardwired to 0 if a device never sets the Relaxed Ordering Attribute in transactions it initiates as a requester. 1h **ENABLE_RELAXED_ORDERING_INIT** (default)
3	R/W	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_UNSUPP_REQ_REPORTING_ENABLE**: This bit enables reporting of Unsupported Requests when set. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. Note: The reporting of error messages (ERR_COR, ERR_NONFATAl, ERR_FATAL) received by Root Port is controlled exclusively by Root Control Register. 0h **UNSUPP_REQ_REPORTING_ENABLE_INIT** (default)
2	R/W	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_FATAL_ERROR_REPORTING_ENABLE**: This bit controls reporting of fatal errors. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. 0h **FATAL_ERROR_REPORTING_ENABLE_INIT** (default)
1	R/W	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_NON_FATAL_ERROR_REPORTING_ENABLE:** This bit controls reporting of non-fatal errors. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. 0h **NON_FATAL_ERROR_REPORTING_ENABLE_INIT** (default)
0	R/W	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_CORR_ERROR_REPORTING_ENABLE**: This bit controls reporting of correctable errors. For a multi-function device, this bit controls error reporting for each function from point-of-view of the respective function. 0h **CORR_ERROR_REPORTING_ENABLE_INIT** (default)

32.2.5.4 T_PCIE2_RP_LINK_CAPABILITIES (x008Ch)

The Link Capabilities Register identifies PCI Express Link specific capabilities.

Link Capabilities Register

Offset: 08ch | Read/Write: RO

Bit	Reset	Description
31:24	None	**T_PCIE2_RP_LINK_CAPABILITIES_PORT_NUMBER**: This field indicates the PCI Express port number for the given PCI Express Link.
23:22	None	**T_PCIE2_RP_LINK_CAPABILITIES_RESERVED**: Reserved.
21	0h	**T_PCIE2_RP_LINK_BW_NOTIFY**: 0h **INIT** (default)

Bit	Reset	Description
20	1h	**T_PCIE2_RP_LINK_CAPABILITIES_LINKACTV_REPORTING:** For a Downstream Port, this bit must be set to 1b if the component supports the optional capability of reporting the DL_Active state of the Data Link Control and Management State Machine. For a hot-plug capable Downstream Port (as indicated by the Hot-Plug Capable field of the Slot Capabilities register), this bit must be set to 1b. 1h LINKACTV_REPORTING_INIT (default)
19	0h	**T_PCIE2_RP_LINK_CAPABILITIES_SURPRISE_DOWN_ERPT_CAP:** For a Downstream Port, this bit must be set to 1b if the component supports the optional capability of detecting and reporting a Surprise Down error condition. 0h SURPRISE_DOWN_ERPT_CAP_INIT (default)
18	0h	**T_PCIE2_RP_LINK_CAPABILITIES_CLOCK_PM**: A value of 1b in this bit indicates that the component tolerates the removal of any reference clock(s) via the "clock request" (CLKREQ#) mechanism when the link is in the L1 and L2/3 Ready link states. A value of 0b indicates the component does not have this capability and that reference clock(s) must not be removed in these link states. This capability is applicable only in form factors that support "clock request" (CLKREQ#) capability. 0h **CLOCK_PM_INIT** (default)
17:15	2h	**T_PCIE2_RP_LINK_CAPABILITIES_L1_EXIT_LATENCY:** This field indicates the L1 exit latency for the given PCI Express Link. The value reported indicates the lenght of time this Port requires to complete transition from L1 to L0. Defined encodings are: • 000b -- Less than 1 us. • 001b -- 1 us to less than 2 us. • 010b -- 2 us to less than 4 us. • 011b -- 4 us to less than 8 us. • 100b -- 8 us to less than 16 us. • 101b -- 16 us to less than 32 us. • 110b -- 32 us-64 us. • 111b -- More than 64 us. 2h L1_EXIT_LATENCY_INIT (default)
14:12	3h	**T_PCIE2_RP_LINK_CAPABILITIES_L0S_EXIT_LATENCY:** This field indicates the L0s exit latency for the given PCI Express Link. The value reported indicates the length of time this Port requires to complete transition from L0s state to L0. Defined encodings are: • 000b -- Less than 64 ns. • 001b -- 64 ns to less than 128 ns. • 010b -- 128 ns to less than 256 ns. • 011b -- 256 ns to less than 512 ns. • 100b -- 512 ns to less than 1 us. • 101b -- 1 us to less than 2 us. • 110b -- 2 us-4us. • 111b -- Reserved. 3h L0S_EXIT_LATENCY_INIT (default)
11:10	11h	**T_PCIE2_RP_LINK_CAPABILITIES_ACTIVE_STATE_LINK_PM_SUPPORT**: This field indicates the level of active state power management supported on the given PCI Express Link. Defined encodings are: • 00b -- Reserved • 01b -- L0s Entry Supported • 10b -- Reserved • 11b -- L0s and L1 Supported 11h ACTIVE_STATE_LINK_PM_SUPPORT_INIT (default)

Bit	Reset	Description
9:4	None	**T_PCIE2_RP_LINK_CAPABILITIES_MAX_LINK_WIDTH:** This field indicates the maximum width of the given PCI Express Link. Defined encodings are: • 000000b -- Reserved • 000001b -- x1 • 000010b -- x2 • 000100b -- x4 • 001000b -- x8 • 001100b -- x12 • 010000b -- x16 • 100000b -- x32
3:0	1h	**T_PCIE2_RP_LINK_CAPABILITIES_MAX_LINK_SPEED:** This field indicates the maximum Link speed of the given PCI Express Link. Defined encodings are: 0001b = 2.5 Gb/s Link All other encodings are reserved. 1h MAX_LINK_SPEED_INIT (default)

32.2.5.5 T_PCIE2_RP_LINK_CONTROL_STATUS (x0090h)

The Link Control register controls PCI Express Link specific parameters. It also provides information about PCI Express Link specific parameters.

Link Control and Link Status Registers

Offset: 090h | Read/Write: R/W

Bit	R/W	Reset	Description
31	RW1C	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_AUTO_BANDWIDTH:** NOTE: this field is reset by Cold Reset 0h **AUTO_BANDWIDTH_TRUE** (default) 1h **AUTO_BANDWIDTH_FALSE** 1h **AUTO_BANDWIDTH_SET**
30	RW1C	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_BW_MANAGEMENT:** NOTE: this field is reset by Cold Reset 0h **BW_MANAGEMENT_TRUE** (default) 1h **BW_MANAGEMENT_FALSE** 1h **BW_MANAGEMENT_SET**
29	R	None	**T_PCIE2_RP_LINK_CONTROL_STATUS_DL_LINK_ACTIVE:** This bit indicates the status of the Data Link Control and Management State Machine. It returns a 1b to indicate the DL_Active state, 0b otherwise. This bit must be implemented if the corresponding Data Link Layer Active Capability bit is implemented. Otherwise, this bit must be hardwired to 0b.
28	R	None	**T_PCIE2_RP_LINK_CONTROL_STATUS_SLOT_CLOCK_CONFIG:** This bit indicates that the component uses the same physical reference clock that the platform provides on the connector. If the device uses an independent clock irrespective of the presence of a reference on the connector, this bit must be clear.
27	R	None	**T_PCIE2_RP_LINK_CONTROL_STATUS_LINK_TRAINING:** This read-only bit indicates that Link training is in progress; hardware clears this bit once training is complete. Note: This field is not applicable and reserved for endpoint devices and Upstream Ports of Switches.
26	R	None	**T_PCIE2_RP_LINK_CONTROL_STATUS_UNDEFINED:** The value read from this bit is undefined. In previous versions of this specification, this bit was used to indicate a Link Training Error. System software must ignore the value read from this bit. System software is permitted to write any value to this bit.

Bit	R/W	Reset	Description
25:20	R	None	**T_PCIE2_RP_LINK_CONTROL_STATUS_NEG_LINK_WIDTH:** This field indicates the negotiated width of the given PCI Express Link. Defined encodings are: • 000001b -- x1 • 000010b -- x2 • 000100b -- x4 • 001000b -- x8 • 001100b -- x12 • 010000b -- x16 • 100000b -- x32
19:16	R	1h	**T_PCIE2_RP_LINK_CONTROL_STATUS_LINK_SPEED:** This field indicates the negotiated Link Speed of the given PCI Express Link. Defined encodings are: 0001b = 2.5 Gb/s PCI Express Link 0010b = 5.0 Gb/s PCI Express Link All other encodings are reserved. 1h LINK_SPEED_GEN1 (default) 2h LINK_SPEED_GEN2 (default)
15:12	R	0	Reserved
11	R/W	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_AUTO_BANDWIDTH_INT_EN:** 0h **AUTO_BANDWIDTH_INT_EN_INIT** (default)
10	R/W	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_BW_MANAGEMENT_INT_EN:** 0h **BW_MANAGEMENT_INT_EN_INIT** (default)
9	R/W	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_HW_AUTO_WIDTH_DISABLE:** 0h **HW_AUTO_WIDTH_DISABLE_DEFAULT** (default)
8	R	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_CLOCK_PM:** **0h CLOCK_PM_DEFAULT (default)**
7	R/W	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_EXTENDED_SYNCH:** This bit when set forces extended transmission of FTS ordered sets in FTS and extra TS2 at exit from L1 prior to entering L0. This mode provides external devices monitoring the link time to achieve bit and symbol lock before the link enters L0 state and resumes communication. Default value for this bit is 0. 0h EXTENDED_SYNCH_INIT (default)
6	R/W	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_COMMON_CLOCK_CONFIGURATION:** This bit when set indicates that this component and the component at the opposite end of this Link are operating with a distributed common reference clock. A value of 0 indicates that this component and the component at the opposite end of this Link are operating with asynchronous reference clock. Components utilize this common clock configuration information to report the correct L0s and L1 Exit Latencies. 0h COMMON_CLOCK_CONFIGURATION_INIT (default)
5	R	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_RETRAIN_LINK:** This bit initiates Link retraining when set. This field is not applicable and reserved for endpoint devices and Upstream Ports of a Switch. This bit always returns 0 when read. 0h RETRAIN_LINK_INIT (default)
4	R/W	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_LINK_DISABLE:** This bit disables the Link when set to 1b. This field is not applicable and reserved for endpoint devices and Upstream Ports of a Switch. Writes to this bit are immediately reflected in the value read from the bit, regardless of the actual Link State. 0h LINK_DISABLE_INIT (default)
3	R	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_READ_COMPLETION_BOUNDARY:** Encodings are: 0b = 64 byte 1b = 128 byte This field is hardwired for a Root Port and returns its RCB support capabilities. Devices that do not implement this feature must hardwire the field to 0b. This field is R/W for devices other than Root Ports. 0h READ_COMPLETION_BOUNDARY_INIT (default)

Bit	R/W	Reset	Description
2	R	0	Reserved
1:0	R/W	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_ACTIVE_STATE_LINK_PM_CONTROL:** This field controls the level of active state PM supported on the given PCI Expres Link. Defined encodings are: • 00b -- Reserved • 01b -- L0s Entry Supported • 10b -- Reserved • 11b -- L0s and L1 Supported 0h ACTIVE_STATE_LINK_PM_CONTROL_INIT (default)

32.2.5.6 T_PCIE2_RP_SLOT_CAPABILITIES (x0094h)

The Slot Capabilities register identifies PCI Express slot specific capabilities.

Slot Capabilities Register

Offset: 094h | Read/Write: RO

Bit	Reset	Description
31:19	None	**T_PCIE2_RP_SLOT_CAPABILITIES_PHYSICAL_SLOT_NUMBER:** This hardware initialized field indicates the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is globally unique within the chassis. These registers should be initialized to 0 for ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Switch device or Root Port.
18	0h	**T_PCIE2_RP_SLOT_CAPABILITIES_NO_CMD_COMPLETED_SUPPORT:** When set to 1, this bit indicates that this slot does not generate software notification when an issued command is completed by the Hot Plug Controller. 0h NO_CMD_COMPLETED_SUPPORT_INIT (default)
17	0h	**T_PCIE2_RP_SLOT_CAPABILITIES_ELECTROMECHANICAL_INTERLOCK_PRESENT:** Indicates that the ElectroMechanical Interlock mechanism is implemented. 0h ELECTROMECHANICAL_INTERLOCK_PRESENT_NO (default)
16:15	None	**T_PCIE2_RP_SLOT_CAPABILITIES_SLOT_POWER_LIMIT_SCALE:** This field Specifies the scale used for the Slot Power Limit Value. Range of values: 00b = 1.0x 01b = 0.1x 10b = 0.01x 11b = 0.001x This register must be implemented if the Slot Implemented bit is set. The default value prior to hardware/firmware initialization is 00b.
14:7	None	**T_PCIE2_RP_SLOT_CAPABILITIES_SLOT_POWER_LIMIT_VALUE:** In combination with the Slot Power Limit Scale value, this field specifies the upper limit on power supplied by slot. Power limit (in Watts) is calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field. This register must be implemented if the Slot Implemented bit is set. The default value prior to hardware/firmware initialization is 0000_0000b.
6	0h	**T_PCIE2_RP_SLOT_CAPABILITIES_HOT_PLUG_CAPABLE:** This bit when set indicates that this slot is capable of supporting Hot-plug operations. 0h HOT_PLUG_CAPABLE_NO (default)
5	0h	**T_PCIE2_RP_SLOT_CAPABILITIES_HOT_PLUG_SURPRISE:** This bit when set indicates that a device present in this slot might be removed from the system without any prior notification. 0h HOT_PLUG_SURPRISE_NO (default)
4	0h	**T_PCIE2_RP_SLOT_CAPABILITIES_POWER_INDICATOR_PRESENT:** This bit when set indicates that a Power Indicator is implemented on the chassis for this slot. 0h POWER_INDICATOR_PRESENT_NO (default)
3	0h	**T_PCIE2_RP_SLOT_CAPABILITIES_ATTENTION_INDICATOR_PRESENT:**

Bit	Reset	Description
		This bit when set indicates that an Attention Indicator is implemented on the chassis for this slot. 0h ATTENTION_INDICATOR_PRESENT_NO (default)
2	0h	**T_PCIE2_RP_SLOT_CAPABILITIES_MRL_SENSOR_PRESENT:** This bit when set indicates that an Attention Indicator is implemented on the chassis for this slot. 0h MRL_SENSOR_PRESENT_NO (default)
1	0h	**T_PCIE2_RP_SLOT_CAPABILITIES_POWER_CONTROLLER_PRESENT:** This bit when set indicates that a Power Controller is implemented for this slot. 0h POWER_CONTROLLER_PRESENT_NO (default)
0	0h	**T_PCIE2_RP_SLOT_CAPABILITIES_ATTENTION_BUTTON_PRESENT:** This bit when set indicates that an Attention Button is implemented on the chassis for this slot. 0h ATTENTION_BUTTON_PRESENT_NO (default)

32.2.5.7 T_PCIE2_RP_SLOT_CONTROL_STATUS (x0098h)

The Slot Control Register controls PCI Express Slot specific parameters. It also provides information about PCI Express Slot specific parameters.

Slot Control and Slot Status Registers

Offset: 098h | Read/Write: R/W

Bit	R/W	Reset	Description
31:25	R	0	Reserved
24	RW1C	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_DL_LAYER_STATE_CHANGED:** When set to 1, this field enables software notification when Data Link Layer Active field is changed 0h DL_LAYER_STATE_CHANGED_INIT (default) 1h DL_LAYER_STATE_CHANGED_SET
23	R	None	**T_PCIE2_RP_SLOT_CONTROL_STATUS_ELECTROMECHANICAL_INTERLOCK_STATE:** This bit when set physically locks the add-in card which the user is trying to remove, till software releases it by resetting the bit. Indicates the current state of the interlock, ie whether the card is electro-mechanically engaged or disengaged 0b : disengaged 1b : engaged Software reads this bit to determine what state the card is in 1h ELECTROMECHANICAL_INTERLOCK_STATE_YES
22	R	None	**T_PCIE2_RP_SLOT_CONTROL_STATUS_PRESENCE_DETECT_STATE:** This bit indicates the presence of a card in the slot. The bit reflects the Presence Detect status determined via an in-band mechanism or via the Present Detect pins as defined in the PCI Express Card Electromechanical Specification. Defined encodings are: 0b = Slot Empty. 1b = Card Present in slot. This register is required to be implemented on all Switch devices and Root Ports. The presence detect field for Switch devices or Root Ports not connected to slots should be hardwired to 1. This register is required if a slot is implemented. 1h PRESENCE_DETECT_STATE_YES
21	R	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_MRL_SENSOR_STATE:** This register reports the status of the MRL sensor if it is implemented. Defined encodings are: 0b = MRL Closed. 1b = MRL Open. 0h MRL_SENSOR_STATE_INIT (default)
20	RW1C	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_COMMAND_COMPLETED:** This bit is set when the hot plug controller completes an issued command. 0h COMMAND_COMPLETED_INIT (default) 1h COMMAND_COMPLETED_SET
19	RW1C	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_PRESENCE_DETECT_CHANGED:** This bit is set when a Presence Detect change is detected. 0h PRESENCE_DETECT_CHANGED_INIT (default) 1h PRESENCE_DETECT_CHANGED_SET

Bit	R/W	Reset	Description
18	RW1C	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_MRL_SENSOR_CHANGED:** This bit is set when a MRL Sensor state change is detected. 0h MRL_SENSOR_CHANGED_INIT (default) 1h MRL_SENSOR_CHANGED_SET
17	RW1C	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_POWER_FAULT_DETECTED:** This bit is set when the Power Controller detects a power fault at this slot. 0h POWER_FAULT_DETECTED_INIT (default) 1h POWER_FAULT_DETECTED_SET
16	RW1C	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_ATTN_BUTTON_PRESSED:** This bit is set when the attention button is pressed. 0h ATTN_BUTTON_PRESSED_INIT (default) 1h ATTN_BUTTON_PRESSED_SET
15:13	R	0	Reserved
12	R/W	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_DL_LAYER_STATE_CHANGED_ENABLE:** This bit is set when the value reported in the Data Link Layer Link Active field of the Link Status register is changed 0h DL_LAYER_STATE_CHANGED_ENABLE_INIT (default)
11	R	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_ELECTROMECHANICAL_INTERLOCK_CONTROL:** indicates that the slot supports electromechanical interlock mechanism 0h ELECTROMECHANICAL_INTERLOCK_CONTROL_INIT (default)
10	R/W	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_POWER_CONTROLLER_CONTROL:** When read this register returns the current state of the Power applied to the slot; when written sets the power state of the slot per the defined encodings. 0b = Power On. 1b = Power Off. 0h POWER_CONTROLLER_CONTROL_INIT (default)
9:8	R/W	1h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_POWER_INDICATOR_CONTROL:** Reads to this register return the current state of the Power Indicator; writes to this register set the Power Indicator. 00b = Reserved 01b = On 10b = Blink 11b = Off Writes to this register also cause the Port to send the appropriate POWER_INDICATOR_* messages. 1h POWER_INDICATOR_CONTROL_INIT (default)
7:6	R/W	3h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_ATTN_INDICATOR_CONTROL:** Reads to this register return the current state of the Attention Indicator; writes to this register set the Attention Indicator. Defined encodings are: 00b = Reserved 01b = On 10b = Blink 11b = Off Writes to this register also cause the Port to send the appropriate ATTENTION_INDICATOR_* messages. 3h ATTN_INDICATOR_CONTROL_INIT (default)
5	R/W	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_HOT_PLUG_INTERRUPT_ENABLE:** This bit when set enables generation of hot plug interrupt on enabled hot plug events. 0h HOT_PLUG_INTERRUPT_ENABLE_INIT (default)
4	R/W	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_COMMAND_COMPLETED_INTERRUPT_ENABLE:** This bit when set enables the generation of hot plug interrupt when a command is completed by the Hot plug controller. 0h COMMAND_COMPLETED_INTERRUPT_ENABLE_INIT (default)
3	R/W	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_PRESENCE_DETECT_CHANGED_ENABLE:** This bit when set enables the generation of hot plug interrupt or wake message on a presence detect changed event. 0h PRESENCE_DETECT_CHANGED_ENABLE_INIT (default)
2	R/W	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_MRL_SENSOR_CHANGED_ENABLE:** This bit when set enables the generation of hot plug interrupt or wake message on a MRL sensor

Bit	R/W	Reset	Description
			changed event. 0h MRL_SENSOR_CHANGED_ENABLE_INIT (default)
1	R/W	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_POWER_FAULT_DETECTED_ENABLE:** This bit when set enables the generation of hot plug interrupt or wake message on a power fault event. 0h POWER_FAULT_DETECTED_ENABLE_INIT (default)
0	R/W	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_ATTN_BUTTON_PRESSED_ENABLE:** This bit when set enables the generation of hot plug interrupt or wake message on an attention button pressed event. 0h ATTN_BUTTON_PRESSED_ENABLE_INIT (default)

32.2.5.8 T_PCIE2_RP_RCR (x009Ch)

The Root Control Register controls PCI Express Root Complex specific parameters. Bit positions 31:4 of this address are reserved.

Root Control Register

Offset: 09ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:4	R	0	Reserved
3	R/W	0h	**T_PCIE2_RP_RCR_PME_INT**: The PME_INT bit is enable bit for generating interrupt upon receipt of a PME message as reflected in the PMESTAT bit of Root Status Register. 0h **PME_INT_DIS** (default)
2	R/W	0h	**T_PCIE2_RP_RCR_SERR_FAT**: The SERR_FAT bit is enable bit for generating System Error if a fatal error (ERR_FATAL) is reported by any of the devices in the hierarchy associated with this Root Port, or by the Root Port itself. 0h **SERR_FAT_DIS** (default)
1	R/W	0h	**T_PCIE2_RP_RCR_SERR_NONFAT**: The SERR_NONFAT bit is enable bit for generating System Error if a non-fatal error (ERR_NONFATAL) is reported by any of the devices in the hierarchy associated with this Root Port, or by the Root Port itself. 0h **SERR_NONFAT_DIS** (default)
0	R/W	0h	**T_PCIE2_RP_RCR_SERR_COR**: The SERR_COR bit is enable bit for generating System Error if a correctable error (ERR_COR) is reported by any of the devices in the hierarchy associated with this Root Port, or by the Root Port itself. 0h **SERR_COR_DIS** (default)

32.2.5.9 T_PCIE2_RP_RSR (x00A0h)

The Root Status Register provides information about PCI Express device specific parameters. Bit positions 31:18 are reserved.

Root Status Register

Offset: 0A0h | Read/Write: R/W

Bit	R/W	Reset	Description
31:18	R	0	Reserved

Bit	R/W	Reset	Description
17	R	None	**T_PCIE2_RP_RSR_PMEPEND**: The PMEPEND bit indicates that another PME is pending when the PMESTAT bit is set. When the PMESTAT bit is cleared by software, the PME is delivered by hardware by setting the PMESTAT bit again and updating the REQID appropriately. The PMEPEND bit is cleared by hardware if no more PMEs are pending.
16	RW1C	0h	**T_PCIE2_RP_RSR_PMESTAT**: The PMESTAT bit indicates that PME was asserted by the requester ID indicated in the PME Requester ID field. Subsequent PMEs are kept pending untill the status register is cleared by software by writing a 1. 0h **PMESTAT_NOT_ACTIVE** (default) 1h **PMESTAT_ACTIVE** 1h **PMESTAT_SET**
15:0	R	None	**T_PCIE2_RP_RSR_REQID**: The REQID field indicates the PCI requester ID of the last PME requestor.

32.2.5.10 T_PCIE2_RP_DEVICE_CAPABILITIES_2 (x00A4h)

Offset: 0A4h | Read/Write: RO

Bit	Reset	Description
31:5	0h	**T_PCIE2_RP_DEVICE_CAPABILITIES_2_RESERVED**: Unused bits in this register. 0h **RESERVED_0** (default)
4	1h	**T_PCIE2_RP_DEVICE_CAPABILITIES_2_CPL_TO_DIS_SUP**: Support disabling the completion timeout mechanism. 1h **CPL_TO_DIS_SUP_0** (default)
3:0	3h	**T_PCIE2_RP_DEVICE_CAPABILITIES_2_CPL_TO_RANGES_SUP**: Completion timeout ranges supported by the rootport. Ranges A and B are currently supported by the PCIe 2.0 design. 3h **CPL_TO_RANGES_SUP_0** (default)

32.2.5.11 T_PCIE2_RP_DEVICE_CONTROL_STATUS_2 (x00A8h)

This register is primarily used for Rootport Completion Timeout values. Bits 31:5 are reserved.

Offset: 0A8h | Read/Write: R/W

Bit	Reset	Description
31:5	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_2_RESERVED:** Unused bits in this register. **0h RESERVED_0 (default)**
4	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_2_CPL_TO_DISABLE:** Disables completion timeout, making the rootport always wait endlessly for a completion to a request. **0h CPL_TO_DISABLE_DEFAULT (default)**

Bit	Reset	Description
3:0	0h	**T_PCIE2_RP_DEVICE_CONTROL_STATUS_2_CPL_TO_VALUE:** This field determines the Range and hence the timeout values for Completions expected from the endpoint. Range A 50 Us to 100 ms (80 Us in design) 1 ms to 10 ms (5 ms in design) Range B 16 ms to 55 ms (40 ms in design) 65 ms to 210 ms (140 ms in design) Default 50 Us to 50 ms (10 ms in design) 1h **CPL_TO_VALUE_RANGE_A_LO** 2h **CPL_TO_VALUE_RANGE_A_HI** 5h **CPL_TO_VALUE_RANGE_B_LO** 6h **CPL_TO_VALUE_RANGE_B_HI** 0h **CPL_TO_VALUE_DEFAULT** (default)

32.2.5.12 T_PCIE2_RP_LINK_CAPABILITIES_2 (x00ACh)

This is a placeholder register and is hardwired to 0. There are no capabilities that require this register.

Offset: 0Ach | Read/Write: RO

Bit	Reset	Description
31:0	0h	**T_PCIE2_RP_LINK_CAPABILITIES_2_BITS**: 0h **BITS_0**

32.2.5.13 T_PCIE2_RP_LINK_CONTROL_STATUS_2 (x00B0h)

Offset: 0B0h | Read/Write: R/W

Bit	R/W	Reset	Description
31:17	R	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_2_RESERVED_STATUS**: 0h **RESERVED_STATUS_DEFAULT** (default)
16	R	None	**T_PCIE2_RP_LINK_CONTROL_STATUS_2_CURRENT_DEEMPHASIS_LEVEL**: 1h **CURRENT_DEEMPHASIS_LEVEL_3P5** 0h **CURRENT_DEEMPHASIS_LEVEL_6**
15:13	R	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_2_RESERVED_CONTROL**: 0h **RESERVED_CONTROL_DEFAULT** (default)
12	R/W	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_2_COMPLIANCE_DEEMPHASIS**: NOTE: This field is reset by Cold Reset. 0h **COMPLIANCE_DEEMPHASIS_INIT** (default)
11	R/W	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_2_COMPLIANCE_SOS**: NOTE: This field is reset by Cold Reset. 0h **COMPLIANCE_SOS_INIT** (default)
10	R/W	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_2_ENTER_MODIFIED_COMPLIANCE**: When this bit is set to 1, the device transmits the modified compliance pattern if the LTSSM enters Polling. Compliance state. Default value is 0b. NOTE: This field is reset by Cold Reset. 0h **ENTER_MODIFIED_COMPLIANCE_INIT** (default)

Bit	R/W	Reset	Description
9:7	R/W	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_2_TRANSMIT_MARGIN**: This field controls the value of the non-deemphasized voltage level at the transmitter pins. Encodings: 000b: 800-1200mV for full swing and 400-600mV for half-swing 001b-010b: Values must be monotonic with a non-zero slope 011b: 200-400mV for full-swing and 100-200mV for half-swing 100b-111b: reserved Default value is 0b. NOTE: This field is reset by Cold Reset. 0h **TRANSMIT_MARGIN_INIT** (default)
6	R	None	**T_PCIE2_RP_LINK_CONTROL_STATUS_2_SELECTABLE_DEEMPHASIS**: When link is operating at 5GT/s speed, selects the level of de-emphasis. This value is initialized by hardware and/or BIOS. Encodings: 1b: -3.5dB 0b: -6dB Default value is 1b.
5	R	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_2_HW_AUTO_SPEED_DISABLE**: Link speed change is disabled if set to 1b. Default value is 0b. 0h **HW_AUTO_SPEED_DISABLE_INIT**
4	R/W	0h	**T_PCIE2_RP_LINK_CONTROL_STATUS_2_ENTER_COMPLIANCE**: If set to 1b forces link to enter compliance mode at the speed indicated by Target Link Speed field. Default value is 0b. NOTE: This field is reset by Cold Reset. 0h **ENTER_COMPLIANCE_INIT** (default)
3:0	R/W	2h	**T_PCIE2_RP_LINK_CONTROL_STATUS_2_TARGET_LINK_SPEED**: This field sets an upper limit on the link operational speed by restricting the values advertised in training sequences. Defined encodings: 0001b 2.5Gb/s (Gen1) 0010b 5Gb/s (Gen2) Other encodings are reserved. Default value is 0. NOTE: This field is reset by Cold Reset. 0h **TARGET_LINK_SPEED_GEN2_DIS** 1h **TARGET_LINK_SPEED_2P5** 2h **TARGET_LINK_SPEED_5P0** (default)

32.2.5.14 T_PCIE2_RP_SLOT_CAPABILITIES_2 (x00B4h)

This is a placeholder register and is hardwired to 0. There are no capabilities that require this register.

Offset: 0B4h | Read/Write: RO

Bit	Reset	Description
31:0	0h	**T_PCIE2_RP_SLOT_CAPABILITIES_2_BITS**: 0h **BITS_0**

32.2.5.15 T_PCIE2_RP_SLOT_CONTROL_STATUS_2 (x00B8h)

This is a placeholder register and is hardwired to 0. There are no capabilities that require this register.

Offset: 0B8h | Read/Write: RO

Bit	Reset	Description
31:0	0h	**T_PCIE2_RP_SLOT_CONTROL_STATUS_2_BITS**: 0h **BITS_0**

32.2.6 Error Reporting Capability Registers

32.2.6.1 T_PCIE2_RP_ERPTCAP (0x100h)

This is the Advanced Error Reporting Enhanced Capability Header register.

Offset: 100h | Read/Write: RO

Bit	R/W	Reset	Description
31:20	R	0h	**T_PCIE2_RP_ERPTCAP_ID:** 0h **ID_NONE** (default) 1h **ID_AER**
19:16	R	0h	**T_PCIE2_RP_ERPTCAP_VERSION:** 0h **VERSION_0** (default) 1h **VERSION_1**
15:0	R	0h	**T_PCIE2_RP_ERPTCAP_NEXT_PTR:** 0h **NEXT_PTR_NONE** (default)

32.2.6.2 T_PCIE2_RP_ERPTCAP_UCERR (0x104h)

This is the Uncorrectable Error Status register.

Offset: 104h | Read/Write: R/W

Bit	R/W	Reset	Description
31:21	R	0	Reserved
20	RW1C	0h	**T_PCIE2_RP_ERPTCAP_UCERR_UNSUP_REQ_ERR**: Unsupported Request Error Status NOTE: this field is reset by Cold Reset 0h **UNSUP_REQ_ERR_FALSE** (default) 1h **UNSUP_REQ_ERR_TRUE** 1h **UNSUP_REQ_ERR_CLEAR**
19	RW1C	0h	**T_PCIE2_RP_ERPTCAP_UCERR_ECRC_ERR**: ECRC Error Status NOTE: this field is reset by Cold Reset 0h **ECRC_ERR_FALSE** (default) 1h **ECRC_ERR_TRUE** 1h **ECRC_ERR_CLEAR**
18	RW1C	0h	**T_PCIE2_RP_ERPTCAP_UCERR_MF_TLP**: Malformed TLP Status NOTE: this field is reset by Cold Reset 0h **MF_TLP_FALSE** (default) 1h **MF_TLP_TRUE** 1h **MF_TLP_CLEAR**

Bit	R/W	Reset	Description
17	RW1C	0h	**T_PCIE2_RP_ERPTCAP_UCERR_RCV_OVFL**: Receiver Overflow Status NOTE: this field is reset by Cold Reset 0h **RCV_OVFL_FALSE** (default) 1h **RCV_OVFL_TRUE** 1h **RCV_OVFL_CLEAR**
16	RW1C	0h	**T_PCIE2_RP_ERPTCAP_UCERR_UNEXP_COMP**: Unexpected Completion Status NOTE: this field is reset by Cold Reset 0h **UNEXP_COMP_FALSE** (default) 1h **UNEXP_COMP_TRUE** 1h **UNEXP_COMP_CLEAR**
15	RW1C	0h	**T_PCIE2_RP_ERPTCAP_UCERR_COMP_ABORT**: Completion Abort Status NOTE: this field is reset by Cold Reset 0h **COMP_ABORT_FALSE** (default) 1h **COMP_ABORT_TRUE** 1h **COMP_ABORT_CLEAR**
14	RW1C	0h	**T_PCIE2_RP_ERPTCAP_UCERR_COMP_TO**: Completion Timeout Status NOTE: this field is reset by Cold Reset 0h **COMP_TO_FALSE** (default) 1h **COMP_TO_TRUE** 1h **COMP_TO_CLEAR**
13	RW1C	0h	**T_PCIE2_RP_ERPTCAP_UCERR_FC_PROTO_ERR**: Flow Control Protocol Error Status NOTE: this field is reset by Cold Reset 0h **FC_PROTO_ERR_DEFAULT** (default) 1h **FC_PROTO_ERR_TRUE** 1h **FC_PROTO_ERR_CLEAR**
12	RW1C	0h	**T_PCIE2_RP_ERPTCAP_UCERR_POS_TLP**: Poisoned TLP Status NOTE: this field is reset by Cold Reset 0h **POS_TLP_FALSE** (default) 1h **POS_TLP_TRUE** 1h **POS_TLP_CLEAR**
11:5	R	0	Reserved
4	RW1C	0h	**T_PCIE2_RP_ERPTCAP_UCERR_DLINK_PROTO_ERR**: Data Link Protocol Error Status NOTE: this field is reset by Cold Reset 0h **DLINK_PROTO_ERR_FALSE** (default) 1h **DLINK_PROTO_ERR_TRUE** 1h **DLINK_PROTO_ERR_CLEAR**
3:1	R	0	Reserved
0	R	0h	**T_PCIE2_RP_ERPTCAP_UCERR_TRAINING_ERR**: Training Error Status 0h **TRAINING_ERR_DEFAULT** (default)

32.2.6.3 T_PCIE2_RP_ERPTCAP_UCERR_MK (0x108h)

This is the Uncorrectable Error Mask register.

Offset: 108h | Read/Write: R/W

Bit	R/W	Reset	Description
31:21	R	0	Reserved
20	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_MK_UNSUP_REQ_ERR**: NOTE: this field is reset by Cold Reset 0h **UNSUP_REQ_ERR_NOT_MASKED** (default) 1h **UNSUP_REQ_ERR_MASKED**
19	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_MK_ECRC_ERR**: NOTE: this field is reset by Cold Reset 0h **ECRC_ERR_NOT_MASKED** (default) 1h **ECRC_ERR_MASKED**
18	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_MK_MF_TLP**: NOTE: this field is reset by Cold Reset 0h **MF_TLP_NOT_MASKED** (default) 1h **MF_TLP_MASKED**
17	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_MK_RCV_OVFL**: NOTE: this field is reset by Cold Reset 0h **RCV_OVFL_NOT_MASKED** (default) 1h **RCV_OVFL_MASKED**
16	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_MK_UNEXP_COMP**: NOTE: this field is reset by Cold Reset 0h **UNEXP_COMP_NOT_MASKED** (default) 1h **UNEXP_COMP_MASKED**
15	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_MK_COMP_ABORT**: NOTE: this field is reset by Cold Reset 0h **COMP_ABORT_NOT_MASKED** (default) 1h **COMP_ABORT_MASKED**
14	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_MK_COMP_TO**: NOTE: this field is reset by Cold Reset 0h **COMP_TO_NOT_MASKED** (default) 1h **COMP_TO_MASKED**
13	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_MK_FC_PROTO_ERR**: NOTE: this field is reset by Cold Reset 0h **FC_PROTO_ERR_NOT_MASKED** (default) 1h **FC_PROTO_ERR_MASKED**
12	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_MK_POS_TLP**: NOTE: this field is reset by Cold Reset 0h **POS_TLP_NOT_MASKED** (default) 1h **POS_TLP_MASKED**
11:5	R	0	Reserved
4	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_MK_DLINK_PROTO_ERR**: NOTE: this field is reset by Cold Reset 0h **DLINK_PROTO_ERR_NOT_MASKED** (default) 1h **DLINK_PROTO_ERR_MASKED**
3:1	R	0	Reserved
0	R	0h	**T_PCIE2_RP_ERPTCAP_UCERR_MK_TRAINING_ERR**: 0h **TRAINING_ERR_DEFAULT** (default)

32.2.6.4 T_PCIE2_RP_ERPTCAP_UCERR_SEVR (0x10Ch)

This is the Correctable Error Status register.

Offset: 10Ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:21	R	0	Reserved
20	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_SEVR_UNSUP_REQ_ERR**: NOTE: this field is reset by Cold Reset 0h **UNSUP_REQ_ERR_NON_FATAL** (default) 1h **UNSUP_REQ_ERR_FATAL**
19	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_SEVR_ECRC_ERR**: NOTE: this field is reset by Cold Reset 0h **ECRC_ERR_NON_FATAL** (default) 1h **ECRC_ERR_FATAL**
18	R/W	1h	**T_PCIE2_RP_ERPTCAP_UCERR_SEVR_MF_TLP**: NOTE: this field is reset by Cold Reset 0h **MF_TLP_NON_FATAL** 1h **MF_TLP_FATAL** (default)
17	R/W	1h	**T_PCIE2_RP_ERPTCAP_UCERR_SEVR_RCV_OVFL**: NOTE: this field is reset by Cold Reset 0h **RCV_OVFL_NON_FATAL** 1h **RCV_OVFL_FATAL** (default)
16	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_SEVR_UNEXP_COMP**: NOTE: this field is reset by Cold Reset 0h **UNEXP_COMP_NON_FATAL** (default) 1h **UNEXP_COMP_FATAL**
15	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_SEVR_COMP_ABORT**: NOTE: this field is reset by Cold Reset 0h **COMP_ABORT_NON_FATAL** (default) 1h **COMP_ABORT_FATAL**
14	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_SEVR_COMP_TO**: NOTE: this field is reset by Cold Reset 0h **COMP_TO_NON_FATAL** (default) 1h **COMP_TO_FATAL**
13	R/W	1h	**T_PCIE2_RP_ERPTCAP_UCERR_SEVR_FC_PROTO_ERR**: NOTE: this field is reset by Cold Reset 0h **FC_PROTO_ERR_NON_FATAL** 1h **FC_PROTO_ERR_FATAL** (default)
12	R/W	0h	**T_PCIE2_RP_ERPTCAP_UCERR_SEVR_POS_TLP**: NOTE: this field is reset by Cold Reset 0h **POS_TLP_NON_FATAL** (default) 1h **POS_TLP_FATAL**
11:5	R	0	Reserved
4	R/W	1h	**T_PCIE2_RP_ERPTCAP_UCERR_SEVR_DLINK_PROTO_ERR**: NOTE: this field is reset by Cold Reset 0h **DLINK_PROTO_ERR_NON_FATAL** 1h **DLINK_PROTO_ERR_FATAL** (default)
3:1	R	0	Reserved
0	R	1h	**T_PCIE2_RP_ERPTCAP_UCERR_SEVR_TRAINING_ERR**: 0h **TRAINING_ERR_NON_FATAL** 1h **TRAINING_ERR_FATAL**

32.2.6.5 T_PCIE2_RP_ERPTCAP_CERR (0x110h)

This is the Correctable Error Status register.

Offset: 110h | Read/Write: R/W

Bit	R/W	Reset	Description
31:14	R	0	Reserved
13	RW1C	0h	**T_PCIE2_RP_ERPTCAP_CERR_MK_ADVISORY_NF**: 0h **ADVISORY_NF_NOT_MASKED** 1h **ADVISORY_NF_MASKED** (default)
12	RW1C	0h	**T_PCIE2_RP_ERPTCAP_CERR_MK_RPLY_TO**: 0h **RPLY_TO_NOT_MASKED** (default) 1h **RPLY_TO_MASKED**
11:9	R	0	Reserved
8	RW1C	0h	**T_PCIE2_RP_ERPTCAP_CERR_MK_RPLY_RLOV**: 0h **RPLY_RLOV_NOT_MASKED** (default) 1h **RPLY_RLOV_MASKED**
7	RW1C	0h	**T_PCIE2_RP_ERPTCAP_CERR_MK_BAD_DLLP**: 0h **BAD_DLLP_NOT_MASKED** (default) 1h **BAD_DLLP_MASKED**
6	RW1C	0h	**T_PCIE2_RP_ERPTCAP_CERR_MK_BAD_TLP**: 0h **BAD_TLP_NOT_MASKED** (default) 1h **BAD_TLP_MASKED**
5:1	R	0	Reserved
0	RW1C	0h	**T_PCIE2_RP_ERPTCAP_CERR_MK_RCV_ERR**: 0h **RCV_ERR_NOT_MASKED** (default) 1h **RCV_ERR_MASKED**

32.2.6.6 T_PCIE2_RP_ERPTCAP_CERR_MK (0x114h)

This is the Correctable Error Mask register.

Offset: 114h | Read/Write: R/W

Bit	R/W	Reset	Description
31:14	R	0	Reserved
13	R/W	1h	**T_PCIE2_RP_ERPTCAP_CERR_MK_ADVISORY_NF**: 0h **ADVISORY_NF_NOT_MASKED** 1h **ADVISORY_NF_MASKED** (default)
12	R/W	0h	**T_PCIE2_RP_ERPTCAP_CERR_MK_RPLY_TO**: 0h **RPLY_TO_NOT_MASKED** (default) 1h **RPLY_TO_MASKED**
11:9	R	0	Reserved
8	R/W	0h	**T_PCIE2_RP_ERPTCAP_CERR_MK_RPLY_RLOV**: 0h **RPLY_RLOV_NOT_MASKED** (default) 1h **RPLY_RLOV_MASKED**
7	R/W	0h	**T_PCIE2_RP_ERPTCAP_CERR_MK_BAD_DLLP**: 0h **BAD_DLLP_NOT_MASKED** (default) 1h **BAD_DLLP_MASKED**

Bit	R/W	Reset	Description
6	R/W	0h	**T_PCIE2_RP_ERPTCAP_CERR_MK_BAD_TLP**: 0h **BAD_TLP_NOT_MASKED** (default) 1h **BAD_TLP_MASKED**
5:1	R	0	Reserved
0	R/W	0h	**T_PCIE2_RP_ERPTCAP_CERR_MK_RCV_ERR**: 0h **RCV_ERR_NOT_MASKED** (default) 1h **RCV_ERR_MASKED**

32.2.6.7 T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL (0x118h)

This is the Advanced Error Capabilities and Control register.

Offset: 118h | Read/Write: R

Bit	R/W	Reset	Description
31:9	R	0	Reserved
8	R/W	0h	**T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL_ECRC_CHK_EN**: NOTE: this field is reset by Cold Reset 0h **ECRC_CHK_EN_FALSE** (default) 1h **ECRC_CHK_EN_TRUE**
7	R	1h	**T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL_ECRC_CHK_CAP**: 1h **ECRC_CHK_CAP_TRUE** (default)
6	R/W	0h	**T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL_ECRC_GEN_EN**: NOTE: this field is reset by Cold Reset 0h **ECRC_GEN_EN_FALSE** (default) 1h **ECRC_GEN_EN_TRUE**
5	R	1h	**T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL_ECRC_GEN_CAP**: 1h **ECRC_GEN_CAP_TRUE** (default)
4:0	R	None	**T_PCIE2_RP_ERPTCAP_ADV_ERR_CAP_CNTL_ERR_PTR**:

32.2.6.8 T_PCIE2_RP_ERPTCAP_HDR_LOG_DW0 (0x11Ch)

This register is 16 bytes. The header is captured such that the fields of the header read by software in the same way the headers are presented in the spec, when the register is read using dword accesses. Therefore, byte 0 of the header is located in byte 3 of the Header Log register, byte 1 of the header is in the byte 2 of the header Log register and so forth.

Offset: 11Ch | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	None	**T_PCIE2_RP_ERPTCAP_HDR_LOG_DW0_0**:

32.2.6.9 T_PCIE2_RP_ERPTCAP_HDR_LOG_DW1 (0x120h)

Offset: 120h | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	None	**T_PCIE2_RP_ERPTCAP_HDR_LOG_DW0_1**:

32.2.6.10 T_PCIE2_RP_ERPTCAP_HDR_LOG_DW2 (0x124h)

Offset: 124h | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	None	**T_PCIE2_RP_ERPTCAP_HDR_LOG_DW0_2**:

32.2.6.11 T_PCIE2_RP_ERPTCAP_HDR_LOG_DW3 (0x128h)

Offset: 128h | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	None	**T_PCIE2_RP_ERPTCAP_HDR_LOG_DW0_3**:

32.2.6.12 T_PCIE2_RP_ERPTCAP_ERR_CMD (0x12Ch)

This is the Root Error Command register.

Offset: 12Ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:3	R	0	Reserved
2	R/W	0h	**T_PCIE2_RP_ERPTCAP_ERR_CMD_FATAL_ERR_RPT_EN**: 0h **FATAL_ERR_RPT_EN_FALSE** (default) 1h **FATAL_ERR_RPT_EN_TRUE**
1	R/W	0h	**T_PCIE2_RP_ERPTCAP_ERR_CMD_NONFATAL_ERR_RPT_EN**: 0h **NONFATAL_ERR_RPT_EN_FALSE** (default) 1h **NONFATAL_ERR_RPT_EN_TRUE**
0	R/W	0h	**T_PCIE2_RP_ERPTCAP_ERR_CMD_COR_ERR_RPT_EN**: 0h **COR_ERR_RPT_EN_FALSE** (default) 1h **COR_ERR_RPT_EN_TRUE**

32.2.6.13 T_PCIE2_RP_ERPTCAP_ERR_STS (0x130h)

This is the Root Error Status register.

Offset: 130h | Read/Write: R/W

Bit	R/W	Reset	Description
31:27	R	None	**T_PCIE2_RP_ERPTCAP_ERR_STS_ADV_ERR_INTR_MSG_NUM**: If this function is allocated more than one MSI interrupt number, this register is required to contain the offset between the base message data and the MSI message that is generated when any of the status bits of this capacbility are set. HW is required to update this field so that it is correct if the number of MSI messages assigned to the device changes. 1h **ADV_ERR_INTR_MSG_NUM_ONE**
26:7	R/W	0	Reserved
6	RW1C	0h	**T_PCIE2_RP_ERPTCAP_ERR_STS_FATAL_RCVD**: Set when one or more fatal uncorrectable error messages have been received. NOTE: this field is reset by Cold Reset 0h **FATAL_RCVD_FALSE** (default) 1h **FATAL_RCVD_TRUE** 1h **FATAL_RCVD_CLEAR**

Bit	R/W	Reset	Description
5	RW1C	0h	**T_PCIE2_RP_ERPTCAP_ERR_STS_NONFATAL_RCVD**: Set when one or more non-fatal uncorrectable error messages have been received. NOTE: this field is reset by Cold Reset 0h **NONFATAL_RCVD_FALSE** (default) 1h **NONFATAL_RCVD_TRUE** 1h **NONFATAL_RCVD_CLEAR**
4	RW1C		**T_PCIE2_RP_ERPTCAP_ERR_STS_FIRST_FATAL_RCVD**: Set when the first uncorrectable error message received is for a fatal error. NOTE: this field is reset by Cold Reset 0h **FIRST_FATAL_RCVD_FALSE** (default) 1h **FIRST_FATAL_RCVD_TRUE** 1h **FIRST_FATAL_RCVD_CLEAR**
3	RW1C		**T_PCIE2_RP_ERPTCAP_ERR_STS_MULT_UNCOR_RCVD**: Set when either a fatal or a non-fatal error is received and UNCOR_RCVD is already set. NOTE: this field is reset by Cold Reset 0h **MULT_UNCOR_RCVD_FALSE** (default) 1h **MULT_UNCOR_RCVD_TRUE** 1h **MULT_UNCOR_RCVD_CLEAR**
2	RW1C		**T_PCIE2_RP_ERPTCAP_ERR_STS_UNCOR_RCVD**: Set when either a fatal or a non-fatal error message is received and this bit is not already set. NOTE: this field is reset by Cold Reset 0h **UNCOR_RCVD_FALSE** (default) 1h **UNCOR_RCVD_TRUE** 1h **UNCOR_RCVD_CLEAR**
1	RW1C		**T_PCIE2_RP_ERPTCAP_ERR_STS_MULT_COR_RCVD**: Set when a correctable error message is received and COR_RCVD is already set. NOTE: this field is reset by Cold Reset 0h **MULT_COR_RCVD_FALSE** (default) 1h **MULT_COR_RCVD_TRUE** 1h **MULT_COR_RCVD_CLEAR**
0	RW1C		**T_PCIE2_RP_ERPTCAP_ERR_STS_COR_RCVD**: Set when a correctable error message is received and this bit is not already set NOTE: this field is reset by Cold Reset 0h **COR_RCVD_FALSE** (default) 1h **COR_RCVD_TRUE** 1h **COR_RCVD_CLEAR**

32.2.6.14 T_PCIE2_RP_ERPTCAP_ERR_ID (0x134h)

This is the Error Source Identification register. This register identifies the source (Requestor ID) of first correctable and uncorrectable (non-fatal/fatal) errors reported in the Root Error Status register.

Offset: 134h | Read/Write: R

Bit	R/W	Reset	Description
31:16	R	None	**T_PCIE2_RP_ERPTCAP_ERR_ID_ERR_COR**: 0h **ERR_COR_DEFAULT** (default)
15:0	R	None	**T_PCIE2_RP_ERPTCAP_ERR_ID_ERR_UNCOR**: 0h **ERR_UNCOR_DEFAULT** (default)

32.2.6.15 T_PCIE2_RP_PRIV_XP_DL_0 (0x494h)

The T_PCIE2_RP_PRIV_XP_DL_0 register controls various configurable registers in the Data Link layer.

Offset: 494h | Read/Write: R/W

Bit	R/W	Reset	Description
31:30	R	0	Reserved
29:19	R/W	0h	**T_PCIE2_RP_PRIV_XP_DL_0_GEN2_REPLAY_TIMER_LIMIT**: Setting this register to a non-zero value will cause the DL to use the programmed value as the replay timer limit instead of the default value which is based on the negotiated width as described in section 3.5.2.1 of the PCI Express Base Specification, Rev 1.0a. The replay timer limit corresponds to the Unadjusted Replay Transmission Latency Limit listed in table 3-4 of the spec, except that it should include the Rx_L0s_Adjustment. This register takes effect in 5.0GT/s speed of operation when T_PCIE2_RP_PRIV_XP_DL_1_GEN2_DL_TIMERS_DISABLE is programmed to 0. It has no effect when T_PCIE2_RP_PRIV_XP_DL_1_GEN2_DL_TIMERS_DISABLE is programmed to 1. 0h **GEN2_REPLAY_TIMER_LIMIT_INIT** (default)
18.10	R/W	0h	**T_PCIE2_RP_PRIV_XP_DL_0_GEN2_ACK_TIMER_LIMIT**: Setting this register to a non-zero value will cause the DL to use the programmed value as the ack timer limit instead of the default value which is based on the negotiated width as described in section 3.5.3.1 of the PCI Express Base Specification, Rev 1.0a. The ack timer limit corresponds to the Unadjusted Ack Transmission Latency Limit listed in table 3-5 of the spec,except that it should include the Tx_L0s_Adjustment, and should *not* include the Internal Delay. This register takes effect in 5.0GT/s speed of operation when T_PCIE2_RP_PRIV_XP_DL_1_GEN2_DL_TIMERS_DISABLE is programmed to 0. It has no effect when T_PCIE2_RP_PRIV_XP_DL_1_GEN2_DL_TIMERS_DISABLE is programmed to 1. 0h **GEN2_ACK_TIMER_LIMIT_INIT** (default)
9:1	R/W	0h	**T_PCIE2_RP_PRIV_XP_DL_0_GEN2_UPDATE_FC_THRESHOLD**: Setting this register to a non-zero value will cause the DL to use the programmed value as the update fc frequency instead of the default value which is based on the negotiated width as described in section 2.6.1.2 of the PCI Express Base Specification, Rev 2.0. This register takes effect in 5.0GT/s speed of operation when T_PCIE2_RP_PRIV_XP_DL_1_GEN2_DL_TIMERS_DISABLE is programmed to 0. It has no effect when T_PCIE2_RP_PRIV_XP_DL_1_GEN2_DL_TIMERS_DISABLE is programmed to 1. 0h **GEN2_UPDATE_FC_THRESHOLD_INIT** (default)
0	R/W	0	**T_PCIE2_RP_PRIV_XP_DL_0_GEN2_DL_TIMERS_DISABLE**: Setting this register enables the Gen2 DL timer settings to take effect in 5.0GT/s speed of operation. 0h **GEN2_DL_TIMERS_DISABLE_INIT** (default)

32.2.6.16 T_PCIE2_RP_PIPE_CTL (0xD5Ch)

This register controls (enabling /disabling) the pipelines added to meet timing on various interfaces. A value of 0 disable the extra pipeline.

Offset: D5Ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:9	R	0	Reserved
8	R/W	1h	**T_PCIE2_RP_PIPE_CTL_UFA2WRR_PWTOP**: 1h **UFA2WRR_PWTOP_INIT** (default)
7	R/W	1h	**T_PCIE2_RP_PIPE_CTL_UBFI2DFI_P2P**: 1h **UBFI2DFI_P2P_INIT** (default)
6	R/W	1h	**T_PCIE2_RP_PIPE_CTL_TXBA2DFI_WR**: 1h **TXBA2DFI_WR_INIT** (default)

Bit	R/W	Reset	Description
5	R/W	1h	**T_PCIE2_RP_PIPE_CTL_CMDQ2UFARB**: 1h **CMDQ2UFARB_INIT** (default)
4	R/W	1h	**T_PCIE2_RP_PIPE_CTL_UBFI2DFI_NTT**: 1h **UBFI2DFI_NTT_INIT** (default)
3	R/W	1h	**T_PCIE2_RP_PIPE_CTL_PCA**: 1h **PCA_INIT** (default)
2	R/W	1h	**T_PCIE2_RP_PIPE_CTL_DFIREQ**: 1h **DFIREQ_INIT** (default)
1	R/W	1h	**T_PCIE2_RP_PIPE_CTL_DFIRSP**: 1h **DFIRSP_INIT** (default)
0	R/W	1h	**T_PCIE2_RP_PIPE_CTL_DFI2UBFI**: 1h **DFI2UBFI_INIT** (default)

32.2.7 Vendor-Defined Registers

32.2.7.1 T_PCIE2_RP_RX_HDR_LIMIT (0xE00h)

These register defines the upstream header logical partition sizes. These buffers reside in UBFI. RXL provides the write interface. The buffers constitute asynchronous boundary of xclk -> ufpci_clk.

Offset: E00h | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R/W	0h	**T_PCIE2_RP_RX_HDR_LIMIT_CPL**: Programs the size of the Completion Header Buffer. This field must be programmed with an even value. 0h **CPL_INIT** (default)
15:8	R/W	0h	**T_PCIE2_RP_RX_HDR_LIMIT_PW**: Programs the size of the Posted Write Header Buffer. When ISO PWs are enabled, this register must be halved, and programmed before both ufpci_reset_ and xreset_. This field must be programmed with an even value. 0h **PW_INIT** (default)
7:0	R/W	0h	**T_PCIE2_RP_RX_HDR_LIMIT_NP**: Programs the size of the Non-Posted (Read and Write) Header Buffer. When ISO NPs are enabled, this register must be halved, and programmed before both ufpci_reset_ and xreset_. This field must be programmed with an even value. 0h **NP_INIT** (default)

32.2.7.2 T_PCIE2_RP_RX_DATA_LIMIT (0xE04h)

These buffers reside in UBFI. RXL provides the write interface. The buffers constitute asynchronous boundary of xclk -> ufpci_clk.

Offset: E04h | Read/Write: R/W

Bit	R/W	Reset	Description
31:28	R	0	Reserved

Bit	R/W	Reset	Description
27:20	R/W	0h	**T_PCIE2_RP_RX_DATA_LIMIT_CPL**: Programs the size of the Completions Data Buffer. This field must be programmed with an even value. 0h **CPL_INIT** (default)
19:8	R/W	0h	**T_PCIE2_RP_RX_DATA_LIMIT_PW**: Programs the size of the Posted Writes Data Buffer. When ISO PWs are enabled, this register must be halved, and programmed before both ufpci_reset_ and xreset_. This field must be programmed with an even value. 0h **PW_INIT** (default)
7:0	R/W	0h	**T_PCIE2_RP_RX_DATA_LIMIT_NP**: Programs the size of the Non-Posted Write Header Buffer. This field must be programmed with an even value. 0h **NP_INIT** (default)

32.2.7.3 T_PCIE2_RP_TX_HDR_LIMIT (0xE08h)

These buffers reside in TXBA. DFI provides the write interface. These buffers constitute an asynchronous boundary of dfpci_clk -> xclk.

Offset: E08h | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R/W	0h	**T_PCIE2_RP_TX_HDR_LIMIT_NP**: Programs the size of the downstream Non Posted Header Buffer. This should be programmed with same value as TX_DATA_LIMIT_NP. 0h **NP_INIT** (default)
15:8	R/W	0h	**T_PCIE2_RP_TX_HDR_LIMIT_PW**: Programs the size of the downstream Posted Writes Header Buffer. 0h **PW_INIT** (default
7:0	R/W	0h	**T_PCIE2_RP_TX_HDR_LIMIT_NP**: Programs the size of the downstream Non Posted Header Buffer. This should be programmed with same value as TX_DATA_LIMIT_NP. 0h **NP_INIT** (default)

32.2.7.4 T_PCIE2_RP_TX_DATA_LIMIT (0xE0Ch)

These buffers reside in TXBA. DFI provides the write interface. These buffers constitute an asynchronous boundary of dfpci_clk -> xclk.

Offset: E0Ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:8	R/W	0h	**T_PCIE2_RP_TX_DATA_LIMIT_PW**: Programs the size of the downstream Posted Writes Data Buffer. 0h **PW_INIT** (default)
7:0	R/W	0h	**T_PCIE2_RP_TX_DATA_LIMIT_NP**: Programs the size of the downstream Non Posted Data Buffer. This should be programmed with same value as TX_HDR_LIMIT_NP. 0h **NP_INIT** (default)

32.2.7.5 T_PCIE2_RP_UFPCI (0xE10h)

This register controls upstream FPCI control and arbitration in UBFI and performance fields.

Offset: E10h | Read/Write: R/W

Bit	R/W	Reset	Description
31:28	R	0	Reserved
27:23	R/W	0h	**T_PCIE2_RP_UFPCI_ISO_WEIGHT**: 0h **ISO_WEIGHT_INIT** (default)
22	R/W	0h	**T_PCIE2_RP_UFPCI_ISO_CONTROL_ENABLE**: 0h **ISO_CONTROL_ENABLE_INIT** (default)
21	R/W	0h	**T_PCIE2_RP_UFPCI_ISOPW2HPISO**: If set, converts posted writes with TC >= tc2isomap to HP_ISO writes instead of ISO writes. 0h **ISOPW2HPISO_INIT** (default)
20	R/W	0h	**T_PCIE2_RP_UFPCI_ISONP2HPISO**: If set, converts nonposted reads with TC >= tc2isomap to HP_ISO reads instead of ISO reads. 0h **ISONP2HPISO_INIT** (default)
19:12	R/W	0h	**T_PCIE2_RP_UFPCI_REQ_PEND_PERIOD**: The following timers are reset when they hit the value in REQ_PEND_PERIOD 1. noniso_timer - feeds into tms*2sm_iso_pending if system_stutter is supported 2. iso_timer - feeds into tms*2sm_noniso_pending if system_stutter is supported 3. coh_timer - feeds into tms*2sm_coh_request_pend 4. noncoh_timer - feeds into tms*2sm_noncoh_request_pend 0h **REQ_PEND_PERIOD_INIT** (default)
11:10	R/W	1h	**T_PCIE2_RP_UFPCI_WRR_GRANT_BURST**: This field controls the length of time for which a particular controller gets arbitration within a TMS. It relates to time multiplexed inter-TMS arbitration for controllers within a TMS. 1h **WRR_GRANT_BURST_INIT** (default)
9:5	R/W	0h	**T_PCIE2_RP_UFPCI_PW_PRI_OVR_COUNT**: If a PW has received this number of grants when the priority was swapped to PW, it's priority will be reset. 0h **PW_PRI_OVR_COUNT_INIT** (default)
4:0	R/W	0h	**T_PCIE2_RP_UFPCI_PW_STARV_COUNT**: If an NP has received this number of grants over PW while there are PW pending, then swap the priority to PW. 0h **PW_STARV_COUNT_INIT** (default)

32.2.7.6 T_PCIE2_RP_MISC0 (0xE14h)

This register controls miscellaneous fields.

Offset: E14h | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	0h	**T_PCIE2_RP_MISC0_SHORT_RXL_TIMER**: This bit is used for simulations only, to test the completion timeout feature in RXL. It shortens the wait period for a completion to 1 Us. 0h **SHORT_RXL_TIMER_INIT** (default)
30	R/W	0h	**T_PCIE2_RP_MISC0_P2P_SMALL_ISA_HOLE**: When set, this bit controls the peer2peer settings on address ranges 64'hA_0000 to 64'hF_FFFF. It is used to omit address ranges A, B, C, D, E and F from being peer2peer. 0h **P2P_SMALL_ISA_HOLE_INIT** (default)

Bit	R/W	Reset	Description
29	R/W	0h	**T_PCIE2_RP_MISC0_P2P_F**: If 0 and P2P_SMALL_ISA_HOLE=1, all transactions falling within ranges 64'hF_0000 and 64'hF_FFFF will have the fpci2ufa_cmd_peer2peer bit equal to 0. Else the fpci2ufa_cmd_peer2peer bit will be set according to TOM1 and TOM2 registers. 0h **P2P_F_INIT** (default)
28	R/W	0h	**T_PCIE2_RP_MISC0_P2P_E**: If 0 and P2P_SMALL_ISA_HOLE=1, all transactions falling within ranges 64'hE_0000 and 64'hE_FFFF will have the fpci2ufa_cmd_peer2peer bit equal to 0. Else the fpci2ufa_cmd_peer2peer bit will be set according to TOM1 and TOM2 registers. 0h **P2P_E_INIT** (default)
27	R/W	0h	**T_PCIE2_RP_MISC0_P2P_D**: If 0 and P2P_SMALL_ISA_HOLE=1, all transactions falling within ranges 64'hD_0000 and 64'hD_FFFF will have the fpci2ufa_cmd_peer2peer bit equal to 0. Else the fpci2ufa_cmd_peer2peer bit will be set according to TOM1 and TOM2 registers. 0h **P2P_D_INIT** (default)
26	R/W	0h	**T_PCIE2_RP_MISC0_P2P_C**: If 0 and P2P_SMALL_ISA_HOLE=1, all transactions falling within ranges 64'hC_0000 and 64'hC_FFFF will have the fpci2ufa_cmd_peer2peer bit equal to 0. Else the fpci2ufa_cmd_peer2peer bit will be set according to TOM1 and TOM2 registers. 0h **P2P_C_INIT** (default)
25	R/W	0h	**T_PCIE2_RP_MISC0_P2P_B**: If 0 and P2P_SMALL_ISA_HOLE=1, all transactions falling within ranges 64'hB_0000 and 64'hB_FFFF will have the fpci2ufa_cmd_peer2peer bit equal to 0. Else the fpci2ufa_cmd_peer2peer bit will be set according to TOM1 and TOM2 registers. 0h **P2P_B_INIT** (default)
24	R/W	0h	**T_PCIE2_RP_MISC0_P2P_A**: If 0 and P2P_SMALL_ISA_HOLE=1, all transactions falling within ranges 64'hA_0000 and 64'hA_FFFF will have the fpci2ufa_cmd_peer2peer bit equal to 0. Else the fpci2ufa_cmd_peer2peer bit will be set according to TOM1 and TOM2 registers. 0h **P2P_A_INIT** (default)
23	R/W	0h	**T_PCIE2_RP_MISC0_NISONC2HPISO**: When set, all upstream Noniso, Non-coherent, Non-peer2peer reads will be upgraded to HPISO Reads. This upgrade is available for all controllers in a TMS if any one of the controllers has iso buffers. 0h **NISONC2HPISO_INIT** (default)
22	R	0	Reserved
21	R/W	1h	**T_PCIE2_RP_MISC0_AUTO_XCLK_FREQ_EN**: When set, the xclk for a TMS will switch dynamically (by HW itself) between 250 and 500. 250 if all root-ports in that TMS are in Gen1. 500 If any root-port in that TMS is in Gen2. This bit is valid only for Root Port 0 of that TMS, and dont care for others. 1h **AUTO_XCLK_FREQ_EN_INIT** (default)
20	R/W	0h	**T_PCIE2_RP_MISC0_RXL_CLEAR_DROP**: Clears the DROP ALL status of the receiver. DROP ALL occurs on fatal errors such as Receiver Overflow and Malformed TLPs. Once set, RXL will not allow any other transactions upstream. 0h **RXL_CLEAR_DROP_INIT** (default)
19:4	R/W	FFh	**T_PCIE2_RP_MISC0_P2P_BURST_SIZE**: Unused/unconnected. FFh **P2P_BURST_SIZE_INIT** (default)
3	R/W	0h	**T_PCIE2_RP_MISC0_ISO_PW_ENABLE**: Enables ISO PW transactions for packets with TC >= TC2ISOMAP, on controllers that have iso buffers. 0h **ISO_PW_ENABLE_INIT** (default)

Bit	R/W	Reset	Description
2	R/W	1h	**T_PCIE2_RP_MISC0_ISO_NP_ENABLE**: Enables ISO NP transactions for packets with TC >= TC2ISOMAP, on controllers that have iso buffers. 0h **ISO_NP_ENABLE_INIT** (default)
1	R/W	0h	**T_PCIE2_RP_MISC0_NATIVE_P2P_ENABLE**: Enables native peer2peer transactions on controllers that support it. Posted Write peer2peer transactions will use a shortcut through the design to enable high performance peer2peer accesses. 0h **NATIVE_P2P_ENABLE_INIT** (default)
0	R/W	0h	**T_PCIE2_RP_MISC0_ENABLE_CLUMPING**: When set, enables unitid clumping on controllers that support it. Controllers that support unitid clumping will resort to using it's unitid+1 and an additional 32 sourcetags when it has exhausted it's allotted use of first 32 sourcetags. 0h **ENABLE_CLUMPING_INIT** (default)

32.2.7.7 T_PCIE2_RP_TXBA0 (0xE18h)

This register is specific to the TXBA sub-unit in PCIe 2.0 and controls: 1. Replay buffer limits 2. Completion merging limits 3. Replay timer control fields

Offset: E18h | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	0h	**T_PCIE2_RP_TXBA0_REPLAY_TIMER_EXPIRY**: When TXBA0_USE_REPLAY_TIMER_OFFSET is 1, the value programmed into TXBA0_REPLAY_TIMER_EXPIRY is added to the replay timer that HW calculates. When this bit is 0, the value in TXBA0_REPLAY_TIMER_EXPIRY is selected (if its non-zero) instead of HW value. 0h **REPLAY_TIMER_EXPIRY_INIT** (default)
15:13	R	0	Reserved
12	R/W	0h	**T_PCIE2_RP_TXBA0_USE_REPLAY_TIMER_OFFSET**: 0h **USE_REPLAY_TIMER_OFFSET_INIT** (default)
11	R/W	0h	**T_PCIE2_RP_TXBA0_CMPL_MERGE_UPTO_64DW**: Similar to above field, but the upper limit is 64 DW. When neither 32DW/64DW is set, nor completion merging is enabled, maximum merging supported is 128DW. 0h **CMPL_MERGE_UPTO_64DW_INIT** (default)
10	R/W	0h	**T_PCIE2_RP_TXBA0_CMPL_MERGE_UPTO_32DW**: When set, split completions that have payloads upto 32DW will be merged. Completions that belong to the same read request, but have payloads greater than 32DW will have one fragment merged upto 32DW, in addition to one or more packets containing the rest of the payload (which will also be merged using the same rule). 0h **CMPL_MERGE_UPTO_32DW_INIT** (default)
9	R/W	1h	**T_PCIE2_RP_TXBA0_CMPL_MERGE_DISABLE**: Disables completion merging. By default, completion merging is disabled. 1h **CMPL_MERGE_DISABLE_INIT** (default)
8:0	R/W	0h	**T_PCIE2_RP_TXBA0_REPLAY_BUF_LIMIT**: This register exists per controller. It decides how much of the replay buffer (which is shared among all the controller in a tms) is used for the controller. Range of values can be anything from 0 (when controller is disabled) to full depth of the replay buffer size (when there are no other controllers). 0h **REPLAY_BUF_LIMIT_INIT** (default)

32.2.7.8 T_PCIE2_RP_TXBA1 (0xE1Ch)

This register is specific to the TXBA sub-unit in PCIe 2.0 and controls: 1. Replay buffer limits 2. Completion merging limits 3. Replay timer control fields

Offset: E1Ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:8	R/W	0	**T_PCIE2_RP_TXBA1_CMPL_MERGE_THRESHOLD**: Completion merging is enabled only if the number of valid completions exceeds this programmed register. This is used to control the bandwidth vs. latency consideration. The maximum meaningful value of this field is the maximum number of outstanding completions (which is also size of header buffer allocated to the controller). 0h **CMPL_MERGE_THRESHOLD_INIT** (default)
7:4	R/W	0h	**T_PCIE2_RP_TXBA1_CM_OVER_PW_BURST**: These fields are inputs into the downstream arbiter. If there are bunch of downstream PW and downstream CM, txba arbiter runs PW_OVER_CM_BURST many PW before switching to CM. Once it starts running completions, it runs CM_OVER_PW_BURST completions before switching back to PW. 0h **CM_OVER_PW_BURST_INIT** (default)
3:0	R	0	Reserved

32.2.7.9 T_PCIE2_RP_FORCEFC (0xE20h)

This register updates FC priority flip threshold. Updated FCs become high priority (equivalent priority to UpdateFC timer expiring) if the number of outstanding credits to return for each type exceeds the set threshold.

If the threshold is set to 0, this feature is disabled.

Offset: E20h | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	0h	**T_PCIE2_RP_FORCEFC_NPD_UNRET_THRESH**: 0h **NPD_UNRET_THRESH_INIT** (default)
23:16	R/W	0h	**T_PCIE2_RP_FORCEFC_NPH_UNRET_THRESH**: 0h **NPH_UNRET_THRESH_INIT** (default)
15:8	R/W	0h	**T_PCIE2_RP_FORCEFC_PWD_UNRET_THRESH**: 0h **PWD_UNRET_THRESH_INIT** (default)
7:0	R/W	0h	**T_PCIE2_RP_FORCEFC_PWH_UNRET_THRESH**: 0h **PWH_UNRET_THRESH_INIT** (default)

32.2.7.10 T_PCIE2_RP_TIMEOUT0 (0xE24h)

This is the LINK LTSSM Timeout 0 register.

Offset: E24h | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	Dh	**T_PCIE2_RP_TIMEOUT0_PAD_SPDCHNG_GEN2**: Amount of time to wait for pads to change speed from Gen1 to Gen2. Measured in units of 1us. Dh **PAD_SPDCHNG_GEN2_INIT** (default)

Bit	R/W	Reset	Description
23:8	R/W	1F4	**T_PCIE2_RP_TIMEOUT0_PAD_PWRUP_CM**: Amount of time to wait for pads to power up (e.g. L1 wake) if common-mode voltage was not maintained during power down (i.e. L1P). Measured in units of 40ns. 1F4h **PAD_PWRUP_CM_INIT** (default)
7:0	R/W	Dh	**T_PCIE2_RP_TIMEOUT0_PAD_PWRUP**: Amount of time to wait for pads to power up (e.g. L1 wake) if common-mode voltage was maintained during power down (i.e. L1). Measured in units of 40ns. Dh **PAD_PWRUP_INIT** (default)

32.2.7.11 T_PCIE2_RP_TIMEOUT1 (0xE28h)

This is the LINK LTSSM Timeout 1 register.

Offset: E28h | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	96h	**T_PCIE2_RP_TIMEOUT1_RCVRY_SPD_UNSUCCESS_EIDLE**: Amount of time to wait in E-Idle after unsuccessful speed change. Measured in units of 40ns. 96h **RCVRY_SPD_UNSUCCESS_EIDLE_INIT** (default)
23:16	R/W	14h	**T_PCIE2_RP_TIMEOUT1_RCVRY_SPD_SUCCESS_EIDLE**: Amount of time to wait in E-Idle after successful speed change. Measured in units of 40ns. 14h **RCVRY_SPD_SUCCESS_EIDLE_INIT** (default)
15:0	R/W	2E8h	**T_PCIE2_RP_TIMEOUT1_PAD_SPDCHNG_GEN1**: Amount of time to wait for pads to change speed from Gen2 to Gen1. Measured in units of 1us. 2E8h **PAD_SPDCHNG_GEN1_INIT** (default)

32.2.7.12 T_PCIE2_RP_PRBS (0xE34h)

This is the PRBS Results register.

Offset: E34h | Read/Write: R

Bit	R/W	Reset	Description
31:16	R	None	**T_PCIE2_RP_PRBS_LOCKED**: Each bit indicates that particular lane has locked onto the incoming PRBS pattern and is beginning to count bit errors. For example, if LOCKED[n] is set, lane n has successfully locked onto the incoming PRBS pattern.
15:0	R	None	**T_PCIE2_RP_PRBS_ERR_COUNT_OVERFLOW**: Each bit indicates number of bit mismatches during PRBS run exceeded 65K for that particular lane. For example, if ERR_COUNT_OVERFLOW[n] is set, lane n registers > 65K bit mismatches.

32.2.7.13 T_PCIE2_RP_PRBS_ERR_COUNT (0xE38h)

This is the PRBS Results register.

Offset: E38h | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R	0h	**T_PCIE2_RP_LANE_PRBS_ERR_COUNT**: Number of bit errors detected in the incoming PRBS stream after achieving PRBS lock for the selected lane programmed in ERR_SELECT register.
15:4	R	0	Reserved

Bit	R/W	Reset	Description
3:0	R/W	0h	**T_PCIE2_RP_LANE_PRBS_ERR_SELECT**: Selects which lane number's Error Count shows up in the ERR_COUNT register 0h **SELECT_INIT** (default)

32.2.7.14 T_PCIE2_RP_EIDLE_INFER_TO_0 (0xE3Ch)

This is an EIDLE inference timeout register.

Offset: E3Ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	500h	**T_PCIE2_RP_EIDLE_INFER_TO_0_RCVRCFG_SUC_SPEED**: Infer E-Idle after this amount of time while in Recovery.RcvrCfg or Recovery. Speed and successful_speed_negotiation = 1. Measured in units of UI (rx symbol times). 500h **RCVRCFG_SUC_SPEED_INIT** (default)
15:0	R/W	80h	**T_PCIE2_RP_EIDLE_INFER_TO_0_L0_LPBK**: Infer E-Idle after this amount of time while in L0 or Loopback. Active Slave. Measured in units of µs. 80h **L0_LPBK_INIT** (default)

32.2.7.15 T_PCIE2_RP_EIDLE_INFER_TO_1 (0xE40h)

This is an EIDLE inference timeout register.

Offset: E40h | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	3E80h	**T_PCIE2_RP_EIDLE_INFER_TO_1_UNSUC_SPEED_GEN2**: Infer E-Idle after this amount of time while in Recovery. Speed and successful_speed_negotiation = 0 and operating in Gen2 Speed. Measured in units of UI (rx symbol times). 3E80h **UNSUC_SPEED_GEN2_INIT** (default)
15:0	R/W	7D0h	**T_PCIE2_RP_EIDLE_INFER_TO_1_UNSUC_SPEED_GEN1**: Infer E-Idle after this amount of time while in Recovery. Speed and successful_speed_negotiation = 0 and operating in Gen1 Speed. Measured in units of UI (rx symbol times). 7D0h **UNSUC_SPEED_GEN1_INIT** (default)

32.2.7.16 T_PCIE2_RP_LTSSM_DBGREG (0xE44h)

This is the LTSSM debug register.

Offset: E44h | Read/Write: R/W

Bit	R/W	Reset	Description
31	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM31**: 0h **LINKFSM31_INIT** (default) 1h **LINKFSM31_CLEAR**
30	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM30**: 0h **LINKFSM30_INIT** (default) 1h **LINKFSM30_CLEAR**
29	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM29**: 0h **LINKFSM29_INIT** (default) 1h **LINKFSM29_CLEAR**

Bit	R/W	Reset	Description
28	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM28**: 0h **LINKFSM28_INIT** (default) 1h **LINKFSM28_CLEAR**
27	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM27**: 0h **LINKFSM27_INIT** (default) 1h **LINKFSM27_CLEAR**
26	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM26**: 0h **LINKFSM26_INIT** (default) 1h **LINKFSM26_CLEAR**
25	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM25**: 0h **LINKFSM25_INIT** (default) 1h **LINKFSM25_CLEAR**
24	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM24**: 0h **LINKFSM24_INIT** (default) 1h **LINKFSM24_CLEAR**
23	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM23**: 0h **LINKFSM23_INIT** (default) 1h **LINKFSM23_CLEAR**
22	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM22**: 0h **LINKFSM22_INIT** (default) 1h **LINKFSM22_CLEAR**
21	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM21**: 0h **LINKFSM21_INIT** (default) 1h **LINKFSM21_CLEAR**
20	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM20**: 0h **LINKFSM20_INIT** (default) 1h **LINKFSM20_CLEAR**
19	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM19**: 0h **LINKFSM19_INIT** (default) 1h **LINKFSM19_CLEAR**
18	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM18**: 0h **LINKFSM18_INIT** (default) 1h **LINKFSM18_CLEAR**
17	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM17**: 0h **LINKFSM17_INIT** (default) 1h **LINKFSM17_CLEAR**
16	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM16**: 0h **LINKFSM16_INIT** (default) 1h **LINKFSM16_CLEAR**
15	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM15**: 0h **LINKFSM15_INIT** (default) 1h **LINKFSM15_CLEAR**
14	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM14**: 0h **LINKFSM14_INIT** (default) 1h **LINKFSM14_CLEAR**
13	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM13**: 0h **LINKFSM13_INIT** (default) 1h **LINKFSM13_CLEAR**

Bit	R/W	Reset	Description
12	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM12**: 0h **LINKFSM12_INIT** (default) 1h **LINKFSM12_CLEAR**
11	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM11**: 0h **LINKFSM11_INIT** (default) 1h **LINKFSM11_CLEAR**
10	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM10**: 0h **LINKFSM10_INIT** (default) 1h **LINKFSM10_CLEAR**
9	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM9**: 0h **LINKFSM9_INIT** (default) 1h **LINKFSM9_CLEAR**
8	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM8**: 0h **LINKFSM8_INIT** (default) 1h **LINKFSM8_CLEAR**
7	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM7**: 0h **LINKFSM7_INIT** (default) 1h **LINKFSM7_CLEAR**
6	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM6**: 0h **LINKFSM6_INIT** (default) 1h **LINKFSM6_CLEAR**
5	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM5**: 0h **LINKFSM5_INIT** (default) 1h **LINKFSM5_CLEAR**
4	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM4**: 0h **LINKFSM4_INIT** (default) 1h **LINKFSM4_CLEAR**
3	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM3**: 0h **LINKFSM3_INIT** (default) 1h **LINKFSM3_CLEAR**
2	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM2**: 0h **LINKFSM2_INIT** (default) 1h **LINKFSM2_CLEAR**
1	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM1**: 0h **LINKFSM1_INIT** (default) 1h **LINKFSM1_CLEAR**
0	RW1C	0h	**T_PCIE2_RP_LTSSM_DBGREG_LINKFSM0**: 0h **LINKFSM0_INIT** (default) 1h **LINKFSM0_CLEAR**

32.2.7.17 T_PCIE2_RP_PRIV_ERRSTS (0xE48h)

This is the Detailed Private Error status register.

Offset: E48h | Read/Write: R/W

Bit	R/W	Reset	Description
31	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_REPLAY_TIMER_EXPIRED_ERR**: 0h **REPLAY_TIMER_EXPIRED_ERR_INIT** (default) 1h **REPLAY_TIMER_EXPIRED_ERR_CLEAR**

Bit	R/W	Reset	Description
30	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_SA_ERR**: 0h **SA_ERR_INIT** (default) 1h **SA_ERR_CLEAR**
29	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_DESKEW_ERR**: 0h **DESKEW_ERR_INIT** (default) 1h **DESKEW_ERR_CLEAR**
28	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_TRAINING_ERR**: 0h **TRAINING_ERR_INIT** (default) 1h **TRAINING_ERR_CLEAR**
27	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_DLLP_CRC_ERR**: 0h **DLLP_CRC_ERR_INIT** (default) 1h **DLLP_CRC_ERR_CLEAR**
26	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_8B10B_ERR**: 0h **8B10B_ERR_INIT** (default) 1h **8B10B_ERR_CLEAR**
25	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_REPLAY_STARTED_ERR**: 0h **REPLAY_STARTED_ERR_INIT** (default) 1h **REPLAY_STARTED_ERR_CLEAR**
24	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_REPLAY_ROLLOVER_ERR**: 0h **REPLAY_ROLLOVER_ERR_INIT** (default) 1h **REPLAY_ROLLOVER_ERR_CLEAR**
23	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_PWH_UPDATE_FC_ERR**: 0h **PWH_UPDATE_FC_ERR_INIT** (default) 1h **PWH_UPDATE_FC_ERR_CLEAR**
22	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_PWH_TOO_MANY_CREDITS_ERR**: 0h **PWH_TOO_MANY_CREDITS_ERR_INIT** (default) 1h **PWH_TOO_MANY_CREDITS_ERR_CLEAR**
21	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_PWD_UPDATE_FC_ERR**: 0h **PWD_UPDATE_FC_ERR_INIT** (default) 1h **PWD_UPDATE_FC_ERR_CLEAR**
20	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_PWD_TOO_MANY_CREDITS_ERR**: 0h **PWD_TOO_MANY_CREDITS_ERR_INIT** (default) 1h **PWD_TOO_MANY_CREDITS_ERR_CLEAR**
19	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_NPH_UPDATE_FC_ERR**: 0h **NPH_UPDATE_FC_ERR_INIT** (default) 1h **NPH_UPDATE_FC_ERR_CLEAR**
18	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_NPH_TOO_MANY_CREDITS_ERR**: 0h **NPH_TOO_MANY_CREDITS_ERR_INIT** (default) 1h **NPH_TOO_MANY_CREDITS_ERR_CLEAR**
17	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_NPD_UPDATE_FC_ERR**: 0h **NPD_UPDATE_FC_ERR_INIT** (default) 1h **NPD_UPDATE_FC_ERR_CLEAR**
16	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_NPD_TOO_MANY_CREDITS_ERR**: 0h **NPD_TOO_MANY_CREDITS_ERR_INIT** (default) 1h **NPD_TOO_MANY_CREDITS_ERR_CLEAR**
15	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_CH_UPDATE_FC_ERR**: 0h **CH_UPDATE_FC_ERR_INIT** (default) 1h **CH_UPDATE_FC_ERR_CLEAR**

Bit	R/W	Reset	Description
14	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_CH_TOO_MANY_CREDITS_ERR**: 0h **CH_TOO_MANY_CREDITS_ERR_INIT** (default) 1h **CH_TOO_MANY_CREDITS_ERR_CLEAR**
13	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_CD_UPDATE_FC_ERR**: 0h **CD_UPDATE_FC_ERR_INIT** (default) 1h **CD_UPDATE_FC_ERR_CLEAR**
12	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_CD_TOO_MANY_CREDITS_ERR**: 0h **CD_TOO_MANY_CREDITS_ERR_INIT** (default) 1h **CD_TOO_MANY_CREDITS_ERR_CLEAR**
11	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_ISOPW_DATA_ERR**: 0h **REC_OVFL_ISOPW_DATA_ERR_INIT** (default) 1h **REC_OVFL_ISOPW_DATA_ERR_CLEAR**
10	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_ISOPW_HDR_ERR**: 0h **REC_OVFL_ISOPW_HDR_ERR_INIT** (default) 1h **REC_OVFL_ISOPW_HDR_ERR_CLEAR**
9	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_ISONP_HDR_ERR**: 0h **REC_OVFL_ISONP_HDR_ERR_INIT** (default) 1h **REC_OVFL_ISONP_HDR_ERR_CLEAR**
8	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_CPL_DATA_ERR**: 0h **REC_OVFL_CPL_DATA_ERR_INIT** (default) 1h **REC_OVFL_CPL_DATA_ERR_CLEAR**
7	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_PW_DATA_ERR**: 0h **REC_OVFL_PW_DATA_ERR_INIT** (default) 1h **REC_OVFL_PW_DATA_ERR_CLEAR**
6	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_NP_DATA_ERR**: 0h **REC_OVFL_NP_DATA_ERR_INIT** (default) 1h **REC_OVFL_NP_DATA_ERR_CLEAR**
5	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_CPL_HDR_ERR**: 0h **REC_OVFL_CPL_HDR_ERR_INIT** (default) 1h **REC_OVFL_CPL_HDR_ERR_CLEAR**
4	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_PW_HDR_ERR**: 0h **REC_OVFL_PW_HDR_ERR_INIT** (default) 1h **REC_OVFL_PW_HDR_ERR_CLEAR**
3	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_REC_OVFL_NP_HDR_ERR**: 0h **REC_OVFL_NP_HDR_ERR_INIT** (default) 1h **REC_OVFL_NP_HDR_ERR_CLEAR**
2	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_FRAMING_ERR**: 0h **FRAMING_ERR_INIT** (default) 1h **FRAMING_ERR_CLEAR**
1	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_SEQ_ERR**: 0h **SEQ_ERR_INIT** (default) 1h **SEQ_ERR_CLEAR**
0	RW1C	0h	**T_PCIE2_RP_PRIV_ERRSTS_LCRC_ERR**: 0h **LCRC_ERR_INIT** (default) 1h **LCRC_ERR_CLEAR**

32.2.7.18 T_PCIE2_RP_PRIV_ERRMSK (0xECh)

This is the Detailed Private Error status register.

Offset: E4Ch | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_REPLAY_TIMER_EXPIRED_ERR_EN**: 1h **REPLAY_TIMER_EXPIRED_ERR_EN_INIT** (default) 0h **REPLAY_TIMER_EXPIRED_ERR_EN_OFF** 1h **REPLAY_TIMER_EXPIRED_ERR_EN_ON**
30	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_SA_ERR_EN**: 1h **SA_ERR_EN_INIT** (default) 0h **SA_ERR_EN_OFF** 1h **SA_ERR_EN_ON**
29:28	R/W	0	Reserved
27	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_DLLP_CRC_ERR_EN**: 1h **DLLP_CRC_ERR_EN_INIT** (default) 0h **DLLP_CRC_ERR_EN_OFF** 1h **DLLP_CRC_ERR_EN_ON**
26	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_8B10B_ERR_EN**: 1h **8B10B_ERR_EN_INIT** (default) 0h **8B10B_ERR_EN_OFF** 1h **8B10B_ERR_EN_ON**
25	R	0	Reserved
24	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_REPLAY_ROLLOVER_ERR_EN**: 1h **REPLAY_ROLLOVER_ERR_EN_INIT** (default) 0h **REPLAY_ROLLOVER_ERR_EN_OFF** 1h **REPLAY_ROLLOVER_ERR_EN_ON**
23	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_PWH_UPDATE_FC_ERR_EN**: 1h **PWH_UPDATE_FC_ERR_EN_INIT** (default) 0h **PWH_UPDATE_FC_ERR_EN_OFF** 1h **PWH_UPDATE_FC_ERR_EN_ON**
22	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_PWD_TOO_MANY_CREDITS_ERR_EN**: 1h **PWD_TOO_MANY_CREDITS_ERR_EN_INIT** (default) 0h **PWD_TOO_MANY_CREDITS_ERR_EN_OFF** 1h **PWD_TOO_MANY_CREDITS_ERR_EN_ON**
21	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_PWD_UPDATE_FC_ERR_EN**: 1h **PWD_UPDATE_FC_ERR_EN_INIT** (default) 0h **PWD_UPDATE_FC_ERR_EN_OFF** 1h **PWD_UPDATE_FC_ERR_EN_ON**
20	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_PWD_TOO_MANY_CREDITS_ERR_EN**: 1h **PWD_TOO_MANY_CREDITS_ERR_EN_INIT** (default) 0h **PWD_TOO_MANY_CREDITS_ERR_EN_OFF** 1h **PWD_TOO_MANY_CREDITS_ERR_EN_ON**
19	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_NPH_UPDATE_FC_ERR_EN**: 1h **NPH_UPDATE_FC_ERR_EN_INIT** (default) 0h **NPH_UPDATE_FC_ERR_EN_OFF** 1h **NPH_UPDATE_FC_ERR_EN_ON**
18	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_NPH_TOO_MANY_CREDITS_ERR_EN**: 1h **NPH_TOO_MANY_CREDITS_ERR_EN_INIT** (default) 0h **NPH_TOO_MANY_CREDITS_ERR_EN_OFF** 1h **NPH_TOO_MANY_CREDITS_ERR_EN_ON**
17	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_NPD_UPDATE_FC_ERR_EN**: 1h **NPD_UPDATE_FC_ERR_EN_INIT** (default) 0h **NPD_UPDATE_FC_ERR_EN_OFF** 1h **NPD_UPDATE_FC_ERR_EN_ON**

Bit	R/W	Reset	Description
16	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_NPD_TOO_MANY_CREDITS_ERR_EN**: 1h **NPD_TOO_MANY_CREDITS_ERR_EN_INIT** (default) 0h **NPD_TOO_MANY_CREDITS_ERR_EN_OFF** 1h **NPD_TOO_MANY_CREDITS_ERR_EN_ON**
15	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_CH_UPDATE_FC_ERR_EN**: 1h **CH_UPDATE_FC_ERR_EN_INIT** (default) 0h **CH_UPDATE_FC_ERR_EN_OFF** 1h **CH_UPDATE_FC_ERR_EN_ON**
14	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_CH_TOO_MANY_CREDITS_ERR_EN**: 1h **CH_TOO_MANY_CREDITS_ERR_EN_INIT** (default) 0h **CH_TOO_MANY_CREDITS_ERR_EN_OFF** 1h **CH_TOO_MANY_CREDITS_ERR_EN_ON**
13	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_CD_UPDATE_FC_ERR_EN**: 1h **CD_UPDATE_FC_ERR_EN_INIT** (default) 0h **CD_UPDATE_FC_ERR_EN_OFF** 1h **CD_UPDATE_FC_ERR_EN_ON**
12	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_CD_TOO_MANY_CREDITS_ERR_EN**: 1h **CD_TOO_MANY_CREDITS_ERR_EN_INIT** (default) 0h **CD_TOO_MANY_CREDITS_ERR_EN_OFF** 1h **CD_TOO_MANY_CREDITS_ERR_EN_ON**
11	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_DLL_PROTOCOL_ERR_EN**: 1h **DLL_PROTOCOL_ERR_EN_INIT** (default) 0h **DLL_PROTOCOL_ERR_EN_OFF** 1h **DLL_PROTOCOL_ERR_EN_ON**
10:3	R	0	Reserved
2	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_FRAMING_ERR_EN**: 1h **FRAMING_ERR_EN_INIT** (default) 0h **FRAMING_ERR_EN_OFF** 1h **FRAMING_ERR_EN_ON**
1	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_SEQ_ERR_EN**: 1h **SEQ_ERR_EN_INIT** (default) 0h **SEQ_ERR_EN_OFF** 1h **SEQ_ERR_EN_ON**
0	R/W	1h	**T_PCIE2_RP_PRIV_ERRMSK_LCRC_ERR_EN**: 1h **LCRC_ERR_EN_INIT** (default) 0h **LCRC_ERR_EN_OFF** 1h **LCRC_ERR_EN_ON**

32.2.7.19 T_PCIE2_RP_LTSSM_TRACE_CONTROL (0xE50h)

This is the LTSSM Trace Control register. Ltssm state changes are stored in a RAM. This register controls storing of this change in the ram.

Offset: E50h | Read/Write: R/W

Bit	R/W	Reset	Description
31:14	R	0	Reserved
13:11	R/W	0h	**T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_PRX_LTSSM_MINOR**: 0h **TRIG_PRX_LTSSM_MINOR_INIT** (default)
10:8	R/W	0h	**T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_PTX_LTSSM_MINOR**: 0h **TRIG_PTX_LTSSM_MINOR_INIT** (default)

Bit	R/W	Reset	Description
7:4	R/W	0h	**T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_LTSSM_MAJOR**: 0h **TRIG_LTSSM_MAJOR_INIT** (default)
3	R/W	0h	**T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_ON_EVENT**: when this bit is set, ltssm changes will start storing only after its state has reached the state specified in T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_LTSSM_MAJOR, T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_PTX_LTSSM_MINOR and T_PCIE2_RP_LTSSM_TRACE_CONTROL_TRIG_PRX_LTSSM_MINOR state. T_PCIE2_RP_LTSSM_ 0h **TRIG_ON_EVENT_INIT** (default)
2	R/W	0h	**T_PCIE2_RP_LTSSM_TRACE_CONTROL_CLEAR_RAM**: When written with 1, will clear all entries in ram. 0h **CLEAR_RAM_INIT** (default)
1	R/W	0h	**T_PCIE2_RP_LTSSM_TRACE_CONTROL_WRAP_EN**: When this bit is set ram is updated every time ltssm state change happens irrespective of whether write pointer has wrapped around. When this bit is clear ram update is stopped when ram is full. Ram need to be cleared by writing to T_PCIE2_RP_LTSSM_TRACE_CONTROL_CLEAR_RAM before it can start storing new trace. 0h **WRAP_EN_INIT** (default) 0h **WRAP_EN_CLEAR** 1h **WRAP_EN_SET**
0	R/W	1h	**T_PCIE2_RP_LTSSM_TRACE_CONTROL_STORE_EN**: This bit controls whether RAM need to be updated whenever ltssm_state change happens. 1h **STORE_EN_INIT** (default) 0h **STORE_EN_CLEAR** 1h **STORE_EN_SET**

32.2.7.20 T_PCIE2_RP_LTSSM_TRACE_STATUS (0xE54h)

This register helps to check the LTSSM trace which has been stored in RAM.

Offset: E54h | Read/Write: R/W

Bit	R/W	Reset	Description
31:22	R	0	Reserved
21:19	R	None	**T_PCIE2_RP_LTSSM_TRACE_STATUS_PRX_LTSSM_MINOR**: This register returns the data prx_ltssm_minor in ltssm_trace RAM point by T_PCIE2_RP_LTSSM_TRACE_STATUS_READ_ADDR
18:16	R	None	**T_PCIE2_RP_LTSSM_TRACE_STATUS_PTX_LTSSM_MINOR**: This register returns the data ptx_ltssm_minor in ltssm_trace RAM point by T_PCIE2_RP_LTSSM_TRACE_STATUS_READ_ADDR
15:12	R	None	**T_PCIE2_RP_LTSSM_TRACE_STATUS_LTSSM_MAJOR**: This register returns the data ltssm_major in ltssm_trace RAM point by T_PCIE2_RP_LTSSM_TRACE_STATUS_READ_ADDR
11	R	None	**T_PCIE2_RP_LTSSM_TRACE_STATUS_READ_DATA_VALID**: This register returns the data_valid in ltssm_trace RAM point by T_PCIE2_RP_LTSSM_TRACE_STATUS_READ_ADDR. Data valid bit is set an entry is written to that location of ram.
10:6	R/W	0h	**T_PCIE2_RP_LTSSM_TRACE_STATUS_READ_ADDR**: This is read address to ltssm trace ram 0h **READ_ADDR_INIT** (default)
5:1	R	None	**T_PCIE2_RP_LTSSM_TRACE_STATUS_WRITE_PTR**: This field indicates the current location of write pointer. This is especially useful when writing to this RAM is enabled even when ram is full and write pointer is allowed to wrap around.

Bit	R/W	Reset	Description
0	R	None	**T_PCIE2_RP_LTSSM_TRACE_STATUS_RAM_FULL**: This field indicates whether all entries in RAM have been written. This is especially useful when writing to this RAM is enabled even when RAM is full and write pointer is allowed to wrap around.

32.2.7.21 T_PCIE2_RP_PG (0xE58h)

This register helps power gate the PCIe TMS in NVIDIA-specific ways.

Offset: E58h | Read/Write: R/W

Bit	R/W	Reset	Description
31:3	R	0	Reserved
2	R/W	0h	**T_PCIE2_RP_PG_RCVD_PME_TO_ACK_INTR_EN**: This field enables generation of interrupt on reception of PME TO ACK TLP. 1h **RCVD_PME_TO_ACK_INTR_EN_ENABLED** 0h **RCVD_PME_TO_ACK_INTR_EN_DISABLED** (default)
1	R/W	0h	**T_PCIE2_RP_PG_RCVD_PME_TO_ACK**: When the root port receives a PME TO ACK TLP, it sets this field. SW can write 1 to this field to clear it to 0. NOTE: this field is reset by Cold Reset 0h **RCVD_PME_TO_ACK_INIT** (default) 1h **RCVD_PME_TO_ACK_CLEAR**
0	R/W	0h	**T_PCIE2_RP_PG_SEND_PME_TO_MSG**: Writing 1 to this field when from previous value of 0 causes the root port to send a PME TO message downstream. When the PME TO message is sent to lower stages of the root port (txtf), this field is automatically cleared to 0 by the root port. 0h **SEND_PME_TO_MSG_INIT** (default) 1h **SEND_PME_TO_MSG_SEND_NOW**

32.2.7.22 T_PCIE2_RP_VAR_RANGE0 (0xE5Ch)

The PCIe Legacy I/O Decode Range Control Registers allow a programmable I/O address range below 4K to be redirected to a PCIe bus downstream from a RP. This is normally not possible due to the definition of the standard I/O Base/Limit registers in the P2P register set. The purpose is to allow legacy devices on PCIe endpoints to operate at their PC legacy default address ranges.

The range is controlled by a pair of generic 12-bit base/limit address register. The ranges can be mapped anywhere in the lower 4K I/O address space. I/O addresses above 4K can be allocated via standard PCI PnP mechanisms.

Offset: E5Ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:18	R/W	0h	**T_PCIE2_RP_VAR_RANGE0_LIMIT**: Top address for the variable range positive decode addresses. This will be set to the highest address in the range that the PCIE RP is to decode. Both this range and the base must be set properly for any cycles to be decoded. LIMIT[1:0] always 11'b 0h **LIMIT_DEFAULT** (default)
17:16	R	0h	**T_PCIE2_RP_VAR_RANGE0_RSVD1**: 0h **RSVD1_ZER0** (default)

Bit	R/W	Reset	Description
15:2	R/W	0h	**T_PCIE2_RP_VAR_RANGE0_BASE**: Base address for the variable range positive decode addresses. This will be set to the lowest address in the range that the PCIE RP is to decode. Both the base and the limit must be set properly for any cycles to be decoded. BASE must be DWORD aligned 0h **BASE_DEFAULT** (default)
1	R	0h	**T_PCIE2_RP_VAR_RANGE0_RSVD0**: 0h **RSVD0_ZER0** (default)
0	R/W	0h	**T_PCIE2_RP_VAR_RANGE0_ENABLE**: If set then positively decode the variable I/O address range defined in T_PCIE2_VAR_RANGE0. 1h **ENABLE_YES** 0h **ENABLE_NO** (default)

32.2.7.23 T_PCIE2_RP_VAR_RANGE1 (0xE60h)

The PCIe Legacy I/O Decode Range Control Registers allow a programmable I/O address range below 4K to be redirected to a PCIe bus downstream from a RP. This is normally not possible due to the definition of the standard I/O Base/Limit registers in the P2P register set. The purpose is to allow legacy devices on PCIe endpoints to operate at their PC legacy default address ranges.

The range is controlled by a pair of generic 12-bit base/limit address register. The ranges can be mapped anywhere in the lower 4K I/O address space. I/O addresses above 4K can be allocated via standard PCI PnP mechanisms.

Offset: E60h | Read/Write: R/W

Bit	R/W	Reset	Description
31:18	R/W	0h	**T_PCIE2_RP_VAR_RANGE1_LIMIT**: Top address for the variable range positive decode addresses. This will be set to the highest address in the range that the PCIe RP is to decode. Both this range and the base must be set properly for any cycles to be decoded. LIMIT[1:0] always 11'b 0h **LIMIT_DEFAULT** (default)
17:16	R	0h	**T_PCIE2_RP_VAR_RANGE1_RSVD1**: 0h **RSVD1_ZER0** (default)
15:2	R/W	0h	**T_PCIE2_RP_VAR_RANGE1_BASE**: Base address for the variable range positive decode addresses. This will be set to the lowest address in the range that the PCIe RP is to decode. Both the base and the limit must be set properly for any cycles to be decoded. BASE must be DWORD aligned 0h **BASE_DEFAULT** (default)
1	R	0h	**T_PCIE2_RP_VAR_RANGE1_RSVD0**: 0h **RSVD0_ZER0** (default)
0	R/W	0h	**T_PCIE2_RP_VAR_RANGE1_ENABLE**: If set then positively decode the variable I/O address range defined in T_PCIE2_RP_VAR_RANGE1. 1h **ENABLE_YES** 0h **ENABLE_NO** (default)

32.2.7.24 T_PCIE2_RP_VEND_XP (0xF00h)

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: F00h | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	0h	**T_PCIE2_RP_VEND_XP_FORCE_COMPLIANCE**: 0h **FORCE_COMPLIANCE_INIT** (default)
30	R	None	**T_PCIE2_RP_VEND_XP_DL_UP**: This read-only bit tells status of Data Link Layer in XP. This bit was added before the DL Link Active Reporting feature was introduced in the official PCI-Express Specification. It asserts when the DL enters the InitFC2 phase of training. This means that DL_UP will assert before the DL_LINK_ACTIVE bit asserts.
29	R/W	0h	**T_PCIE2_RP_VEND_XP_INTERLEAVE_DLLPS**: Setting this bit will cause the DL to schedule Ack and UpdateFC packets earlier or later than their respective timer expiration times in order to decrease the likelihood of losing efficiency by sending two DLLPs consecutively. 0h **INTERLEAVE_DLLPS_INIT** (default)
28	R/W	0h	**T_PCIE2_RP_VEND_XP_OPPORTUNISTIC_UPDATEFC**: If this bit is set, the DL will send any pending UpdateFC packet whenever it has nothing else to send, instead of waiting for the UpdateFC timer to expire. 0h **OPPORTUNISTIC_UPDATEFC_INIT** (default)
27	R/W	0h	**T_PCIE2_RP_VEND_XP_OPPORTUNISTIC_ACK**: If this bit is set, the DL will send pending Acks whenever it has nothing else to send, instead of waiting for the Ack Timer to expire. 0h **OPPORTUNISTIC_ACK_INIT** (default)
26	R/W	0h	**T_PCIE2_RP_VEND_XP_TRAIN_ERR_ENABLE**: Enables reporting of training errors. 0h **TRAIN_ERR_ENABLE_INIT** (default)
25:18	R/W	0h	**T_PCIE2_RP_VEND_XP_UPDATE_FC_THRESHOLD**: This field specifies an override for the UpdateFC frequency. Setting this field to a non-zero value will cause the TL to use the programmed value MULTIPLIED BY TWO as the UpdateFC timer limit instead of the default value which is based on the negotiated width as described in section 2.6.1.2 of the PCI Express Base Specification, Rev 1.0a. The UpdateFC timer limit corresponds to the UpdateFC Transmission Latency Limit listed in table 2-28 of the specification, except that it should *not* include the InternalDelay. 0h **UPDATE_FC_THRESHOLD_INIT** (default)
17:2	R	None	**T_PCIE2_RP_VEND_XP_PRBS_STAT**: This field returns the results of loopback mode testing. Each bit represents the status of one lane (1 == PASS).
1	R/W	0h	**T_PCIE2_RP_VEND_XP_PRBS_EN**: Enables the root port as loopback master using PRBS-23 as the test pattern. 0h **PRBS_EN_DISABLED** (default) 1h **PRBS_EN_ENABLED**
0	R/W	0h	**T_PCIE2_RP_VEND_XP_EMULATION**: Enables emulation mode operation. 0h **EMULATION_OFF** (default) 1h **EMULATION_ON**

32.2.7.25 T_PCIE2_RP_VEND_XP1 (0xF04h)

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: F04h | Read/Write: R/W

Bit	R/W	Reset	Description
31:29	R/W	2h	**T_PCIE2_RP_VEND_XP1_L1_EXIT_LATENCY**: This field indicates the L1 exit latency for the given PCI Express Link. The value reported indicates the lenght of time this Port requires to complete transition from L1 to L0. The value is reflected in the LINK_CAPABILITIES register. Defined encodings are: • 000b Less than 1 µs. • 001b 1 us to less than 2 µs. • 010b 2 us to less than 4 µs. • 011b 4 us to less than 8 µs. • 100b 8 us to less than 16 µs. • 101b 16 us to less than 32 µs. • 110b 32 us-64 µs. • 111b More than 64 µs. 2h **L1_EXIT_LATENCY_INIT** (default) 0h **L1_EXIT_LATENCY_LT_1** 1h **L1_EXIT_LATENCY_LT_2** 2h **L1_EXIT_LATENCY_LT_4** 3h **L1_EXIT_LATENCY_LT_8** 7h **L1_EXIT_LATENCY_GT_64**
28	R/W	0h	**T_PCIE2_RP_VEND_XP1_FORCE_DOWNSTREAM_NO_SNOOP**: When this bit is set and ENABLE_NO_SNOOP bit is also set, NO_SNOOP bit is set on all downstream TLPs (except for IO and cfg, for which NO_SNOOP is never set) 1h **FORCE_DOWNSTREAM_NO_SNOOP_YES** 0h **FORCE_DOWNSTREAM_NO_SNOOP_NO** (default)
27	R/W	0h	**T_PCIE2_RP_VEND_XP1_FORCE_UPSTREAM_NONCOH**: When set this bit causes all upstream memory traffic (except MSI) to be non-coherent. 0h **FORCE_UPSTREAM_NONCOH_INIT** (default) 1h **FORCE_UPSTREAM_NONCOH_ENABLED** 0h **FORCE_UPSTREAM_NONCOH_DISABLED**
26:19	R/W	0h	**T_PCIE2_RP_VEND_XP1_LINK_PVT_CTL**: bit[0] : ACK_L1_NO_WAIT Enable behavior described in the proposed errata to 5.3.2.1 of 1.0a specification. If this bit is set to 1 on an upstream component, that component will not wait to accumulate the minimum number of credits required to send the largest possible packet for any FC type before sending PM_Request_Ack DLLPs downstream. After receiving a PM_Enter_L1 DLLP, it will begin sending PM_Request_Ack DLLPs as soon as it has received acknowledgement for the last TLP that has been sent. bit[1] : L23_READY_NO_D3 Enable entry into the L2/3 Ready state when the component is not in the D3 PMCSR state. If this bit is set to 0, the component will not allow the link to enter the L2/3 Ready state if it is not in D3. If the bit is set to 1, the link will enter L2/3 Ready after the component receives a PME_Turn_Off message, regardless of the PMCSR state. bit[2] : L1_ASPM_SUPPORT This bit determines the reported value of L1 ASPM support in the link capability register (ACTIVE_STATE_LINK_PM_SUPPORT, bit 11). bit[3] : DONT_MERGE_PMASNAK With the default setting for this bit (0), a group of PM_Active_State_Request_L1 DLLPs from the endpoint will cause a single PM_ASPM_Nak TLP to be sent. If this bit is set to 1, multiple PM_ASPM_Nak TLP messages will be sent, possibly as many as one PM_ASPM_Nak TLP for each PM_Active_State_Request_L1 DLLP. bit[4] : IGNORE_L0S This bit overrides the L0s setting for the Active Sate PM Control register. If bit[4] == 0 (do not override), then bit 0 of the T_PCIE2_RP_LINK_CONTROL_STATUS register is used to determine if our Tx side drivers should enter L0s or not. If bit[4] == 1 (override), then we ignore bit 0 of T_PCIE2_RP_LINK_CONTROL_STATUS, and never enter Tx.L0s. bit[7] : NP_REQ_TIMEOUT If set, disable downstream NP request timeout. This is for debug purposes when we want to hang instead of timeout. 0h **LINK_PVT_CTL_INIT** (default)

Bit	R/W	Reset	Description
18:10	R/W	0h	**T_PCIE2_RP_VEND_XP1_ACK_TIMER_LIMIT**: This field overrides the Ack Timer Limit for this root port. Setting this field to a non-zero value will cause the DL to use the programmed value for the Ack Timer Limit instead of the default value, which is based on the negotiated width as described in section 3.5.3.1 of the PCI Express Base Specification, Rev 1.0a. The Ack Timer Limit corresponds to the Unadjusted Ack Transmission Latency Limit listed in table 3-5 of the specification, except that it should include the Tx_L0s_Adjustment, and should *not* include the InternalDelay. 0h **ACK_TIMER_LIMIT_INIT** (default)
9	R	0	Reserved
8	R/W	1h	**T_PCIE2_RP_VEND_XP1_RNCTRL_GEN2_WAIT_FOR_FIRST_EIES**: When this bit is programmed to 1, in the Config.LinkWidth.Start state, the inactive lanes that are activated will wait for the next EIEOS transmission before sending the first packet. If programmed to 0, it will not wait for the next EIEOS transmission before sending the first packet. 1h **RNCTRL_GEN2_WAIT_FOR_FIRST_EIES_INIT** (default)
7	R	0h	**T_PCIE2_RP_VEND_XP1_RNCTRL_EN**: When this bit is programmed to a 1, it triggers the dynamic link width re-negotiation procedure in XP. This bit is a constant and always returns a value of 0 on a read. 0h **RNCTRL_EN_ZERO** (default)
6	R/W	1h	**T_PCIE2_RP_VEND_XP1_RNCTRL_GEN2_LINK_UPGRADE**: When this bit is programmed to 1, it selects the 2.0-compliant link width upgrade protocol. If programmed to 0, it selects the NV-proprietary link width upgrade protocol. 1h **RNCTRL_GEN2_LINK_UPGRADE_INIT** (default)
5:0	R/W	10h	**T_PCIE2_RP_VEND_XP1_RNCTRL_MAXWIDTH**: This field inidcates the maximum link width required at the end of dynamic link width re-negotiation process. The default value is 16. 10h **RNCTRL_MAXWIDTH_INIT** (default)

32.2.7.26 T_PCIE2_RP_VEND_XP2 (0xF08h)

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: F08h | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	0h	**T_PCIE2_RP_VEND_XP2_L0S_UPDATE_WAKE**: Setting this field to a non-zero value will cause the DL to wake the PL out of L0s in advance of sending it an UpdateFC packet. When the DL sees that there are fewer cycles left in the UpdateFC timer than the value in the L0S_UPDATE_WAKE register, it will tell the PL to wake out of L0s. 0h **L0S_UPDATE_WAKE_INIT** (default)
23:18	R	0	Reserved
17:8	R/W	3FFh	**T_PCIE2_RP_VEND_XP2_L0S_THRESHOLD**: This field controls the idle time required for TxL0s entry from L0, in symbol times (250 MHz clocks). Once the XP detects that it has no more DLLPs/TLPs to send, it will insert precisely (L0S_THRESHOLD + 1) logical idles between the END symbol of the last DLLP/TLP and the COM of the IDL Ordered Set. If L0S_THRESHOLD == 0x3FF, the L0s entry latency is instead (endpoint N_FTS)*4 symbol times, where (endpoint N_FTS) is the N_FTS value advertised by the endpoint during training. Therefore, the threshold can be set to anywhere from 1 to 1023 symbol times (4ns to 4092ns). 3FFh **L0S_THRESHOLD_INIT** (default) 3FFh **L0S_THRESHOLD_REMOTE_NFTS**
7:0	R/W	0h	**T_PCIE2_RP_VEND_XP2_L0S_ACK_WAKE**: Setting this field to a non-zero value will cause the DL to wake the PL out of L0s in advance of sending it an Ack packet. When the DL sees that there are fewer cycles left in the Ack timer than the value in the L0S_ACK_WAKE register, it will tell the PL to wake out of L0s. 0h **L0S_ACK_WAKE_INIT** (default)

32.2.7.27 T_PCIE2_RP_VEND_XV_TIMEOUT (0xF0Ch)

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: F0Ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	0300 00FAh	Reserved

32.2.7.28 T_PCIE2_RP_VEND_SLOT_STRAP (0xF20h)

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

This is a back-door register used to set various bits in the Slot Capabilities Register, which is part of the PCI-Express Capability Structure, as defined by the PCI-Express Specification. The bits listed here map 1:1 with the bits in the Slot Capabilities Register.

NOTE: Bits marked "Reserved" may actually be writable in some chips. Therefore writes to this register must not change the default POR values for the bits marked "Reserved".

Offset: F20h | Read/Write: R/W

Bit	R/W	Reset	Description
31:19	R/W	0h	**T_PCIE2_RP_VEND_SLOT_STRAP_PHYSICAL_SLOT_NUMBER**: The physical numbering of the slots is left up to the board designer. The board designer must communicate the physical slot numbers to the SBIOS developer as well. 0h **PHYSICAL_SLOT_NUMBER_INIT** (default)
18:17	R	0	Reserved
16:15	R/W	0h	**T_PCIE2_RP_VEND_SLOT_STRAP_SLOT_POWER_LIMIT_SCALE**: This field must be programmed according to the unique power-rail capabilities of each board design. Board designers are responsible for communicating the power limits of each slot to the SBIOS developers. 0h **SLOT_POWER_LIMIT_SCALE_INIT** (default)
14:7	R/W	0h	**T_PCIE2_RP_VEND_SLOT_STRAP_SLOT_POWER_LIMIT_VALUE**: This field must be programmed according to the unique power-rail capabilities of each board design. Board designers are responsible for communicating the power limits of each slot to the SBIOS developers. 0h **SLOT_POWER_LIMIT_VALUE_INIT** (default)
6:0	R	0	Reserved

32.2.7.29 T_PCIE2_RP_VEND_CTL1 (0xF48h)

This register contains fields unique to NVIDIA's implementation of PCI Express devices. This is a control register for general feature and function control.

Offset: F48h | Read/Write: R/W

Bit	R/W	Reset	Description
31:22	R	0	Reserved
21	R/W	1h	**T_PCIE2_RP_VEND_CTL1_POLLING_RESET_FIX_EN**: 1h **POLLING_RESET_FIX_EN_INIT** (default) 1h **POLLING_RESET_FIX_EN_ENABLE** 0h **POLLING_RESET_FIX_EN_DISABLE**

Bit	R/W	Reset	Description
20	R/W	1h	**T_PCIE2_RP_VEND_CTL1_HOTPLUG_IN_TRAFFIC_EN**: 1h **HOTPLUG_IN_TRAFFIC_EN_INIT** (default) 1h **HOTPLUG_IN_TRAFFIC_EN_ENABLE** 0h **HOTPLUG_IN_TRAFFIC_EN_DISABLE**
19	R/W	0h	**T_PCIE2_RP_VEND_CTL1_NISO2ISO**: When set to 1, all upstream non-iso-pw and non-iso-np will be upgraded to ISO. 0h **NISO2ISO_INIT** (default) 1h **NISO2ISO_ENABLE** 0h **NISO2ISO_DISABLE**
18	R/W	0h	**T_PCIE2_RP_VEND_CTL1_ALLOW_UPSTREAM_CMPL_OVERTAKE_PW**: When set to 1, UBFI allows cpl bypass pw regardless of RO bit 0h **ALLOW_UPSTREAM_CMPL_OVERTAKE_PW_DIS** (default) 1h **ALLOW_UPSTREAM_CMPL_OVERTAKE_PW_EN**
17	R/W	0h	**T_PCIE2_RP_VEND_CTL1_SPLIT_ARB_BLOCK_COH**: When set to 0, take the single ufpci2fpci_arb_block_coherent as input. When set to 1, take the separate inputs of ufa2fpci_arb_block_coherent_pw and ufa2fpci_arb_block_coherent_np. 0h **SPLIT_ARB_BLOCK_COH_INIT** (default) 0h **SPLIT_ARB_BLOCK_COH_DIS** 1h **SPLIT_ARB_BLOCK_COH_EN**
16	R/W	0h	**T_PCIE2_RP_VEND_CTL1_HIDE_MSIMAP**: 0h **HIDE_MSIMAP_DIS** (default) 1h **HIDE_MSIMAP_EN**
15	R/W	1h	**T_PCIE2_RP_VEND_CTL1_LINKACTV_REPORTING**: This is a backdoor bit for setting the Data Link Layer Link Active Reporting Capable bit (bit 20 of the Link Capabilities Register). This bit *MUST* be set whenever hot-plug is enabled. Otherwise, it can be set as desired. 1h **LINKACTV_REPORTING_CAPABLE** (default) 0h **LINKACTV_REPORTING_NOT_CAPABLE**
14	R/W	1h	**T_PCIE2_RP_VEND_CTL1_P2P_ISO2NISO**: If set, forces upstream peer-to-peer ISO requests to be peer-to-peer NONISO. 1h **P2P_ISO2NISO_EN** (default) 0h **P2P_ISO2NISO_DIS**
13	R/W	0h	**T_PCIE2_RP_VEND_CTL1_ERPT**: This bit, when 0, hides the entire AER Capability structure from the capabilities list. It causes the "next pointer" of the previous capability to point to NULL. Normally, AER is the last capability in the list. 1h **ERPT_EN** 0h **ERPT_DIS** (default)
12	R/W	0h	**T_PCIE2_RP_VEND_CTL1_HIDE_MSI_CAP**: This bit, when 1, hides the entire MSI Capability structure from the capabilities list. It causes the "next pointer" of the previous capability to point to the capability that comes after MSI. 0h **HIDE_MSI_CAP_DIS** (default) 1h **HIDE_MSI_CAP_EN**
11:5	R	0	Reserved
4	R/W	0h	**T_PCIE2_RP_VEND_CTL1_ACCEPT_MSGD1**: When set, allows acceptance of NVIDIA specific vendor type message with data. 0h **ACCEPT_MSGD1_DIS** (default) 1h **ACCEPT_MSGD1_EN**

Bit	R/W	Reset	Description
3:1	R/W	7h	**T_PCIE2_RP_VEND_CTL1_TC2ISO_MAP**: This field specifies TC2ISO_MAP[2:0]. Upstream traffic with TC[2:0] >= TC2ISO_MAP[2:0] are sent as ISO requests on UFPCI Since TC[2:0] == 0 can never be sent to ISO channel, TC2ISO_MAP[2:0] == 0 is used to force ALL traffic to NONISO 7h **TC2ISO_MAP_7** (default)
0	R/W	0h	**T_PCIE2_RP_VEND_CTL1_P2P_BLOCK_P2P_ONLY**: When set, ignores ufa2fpci_arb_block_coherent for upstream peer2peer requests. 1h **P2P_BLOCK_P2P_ONLY_EN** 0h **P2P_BLOCK_P2P_ONLY_DIS** (default)

32.2.7.30 T_PCIE2_RP_VEND_XP_BIST (0xF4Ch)

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

IOBIST and characterization control register.

Offset: F4Ch | Read/Write: R/W

Bit	R/W	Reset	Description
31	R	0	Reserved
30	R/W	0h	**T_PCIE2_RP_VEND_XP_BIST_GOTO_DETECT_ON_SURPRISE_EIDLE**: 0h **GOTO_DETECT_ON_SURPRISE_EIDLE_INIT** (default)
29	R/W	0h	**T_PCIE2_RP_VEND_XP_BIST_ENABLE_SERR_REPORTING**: 0h **ENABLE_SERR_REPORTING_INIT** (default)
28	R/W	0h	**T_PCIE2_RP_VEND_XP_BIST_GOTO_L1_L2_AFTER_DLLP_DONE**: 0h **GOTO_L1_L2_AFTER_DLLP_DONE_INIT** (default)
27	R/W	0h	**T_PCIE2_RP_VEND_XP_BIST_CTRL_IGNORE_LPBK_EXIT**: 0h **CTRL_IGNORE_LPBK_EXIT_INIT** (default)
26	R/W	0h	**T_PCIE2_RP_VEND_XP_BIST_CTRL_RELAX_LPBK_ENTRY**: 0h **CTRL_RELAX_LPBK_ENTRY_INIT** (default)
25	R/W	0h	**T_PCIE2_RP_VEND_XP_XP_BIST_CTRL_FORCE_PRBS_RST**: 0h **INIT** (default)
24	R/W	0h	**T_PCIE2_RP_VEND_XP_FORCE_DEEMPHASIS_ADVERTISED_EN**: 0h **INIT** (default)
23	R	0h	Reserved
22	R/W	0h	**T_PCIE2_RP_VEND_XP_FORCE_RECEIVER_COMPLIANCE_EN**: 0h **INIT** (default)
21	R/W	0h	**T_PCIE2_RP_VEND_XP_FORCE_COMPLIANCE_GEN2_SPEED_EN**: 0h **INIT** (default)
20	R/W	0h	**T_PCIE2_RP_VEND_XP_BIST_FORCE_COMPLIANCE_AND_ADVERTISE_MODE**: 0h **FORCE_COMPLIANCE_AND_ADVERTISE_MODE_INIT** (default)
19:0	R	0	Reserved

32.2.7.31 T_PCIE2_RP_VEND_XP_FTS (0xF54h)

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: F54h | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	80h	**T_PCIE2_RP_VEND_XP_FTS_TS_DETECT_START**: 80h **TS_DETECT_START_INIT** (default)
23:16	R/W	40h	**T_PCIE2_RP_VEND_XP_FTS_FTS_DETECT_START**: This field represents the number of symbol times we wait before the root port begins looking at FTS ordered sets when exiting L0s -- the incoming data is essentially ignored during this time. 40h **FTS_DETECT_START_INIT** (default)
15:8	R	None	**T_PCIE2_RP_VEND_XP_FTS_N_FTS_REMOTE**: NV_XVE_PRIV_XP_N_FTS_REMOTE represents the N_FTS value advertised by the remote device as recorded from the N_FTS symbol of the received TS ordered sets.
7:0	R/W	1Fh	**T_PCIE2_RP_VEND_XP_FTS_N_FTS**: This field represents the N_FTS value that we advertise to the device on the other side of the link. NOTE: If this field is modified from its default (POR) value, the Replay Timer Limit in T_PCIE2_RP_VEND_XP1 must be adjusted accordingly. 1Fh **N_FTS_INIT** (default)

32.2.7.32 T_PCIE2_RP_VEND_XP_STATS0 (0xF58h)

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: F58h | Read/Write: R/W

Bit	R/W	Reset	Description
31:29	R	0	Reserved
28	R/W	0h	**T_PCIE2_RP_VEND_XP_STATS0_FAILED_L0S_EXITS_INF**: When set to 1, the failed L0s exits counter accumulates indefinitely (i.e. the one millisecond timer is ignored). 0h **FAILED_L0S_EXITS_INF_INIT** (default)
27:25	R	0	Reserved
24	R	0h	**T_PCIE2_RP_VEND_XP_STATS0_FAILED_L0S_EXITS_LC**: Loads the current value from the failed L0s exits counter into the VEND_XP_STATS1 register, and then clears the failed L0s exits counter. The counter begins counting again immediately after it is cleared. Write a 1 to this field to trigger the load-and-clear operation. 0h **FAILED_L0S_EXITS_LC_INIT** (default)
23:21	R	0	Reserved
20	r/w	0h	**T_PCIE2_RP_VEND_XP_STATS0_NAKS_RCVD_INF**: When set to 1, the NAKs received counter accumulates indefinitely (i.e. the 64 microsecond timer is ignored). 0h **NAKS_RCVD_INF_INIT** (default)
19:17	R	0	Reserved
16	R	0h	Loads the current value from the NAKs received counter into the VEND_XP_STATS1 register, and then clears the NAKs received counter. The counter begins counting again immediately after it is cleared. Write a 1 to this field to trigger the load-and-clear operation. 0h **NAKS_RCVD_LC_INIT** (default)
15:13	R	0	Reserved
12	R/W	0h	**T_PCIE2_RP_VEND_XP_STATS0_CRC_ERRORS_INF**: When set to 1, the CRC error counter accumulates indefinitely (i.e. the 64 microsecond timer is ignored). 0h **CRC_ERRORS_INF_INIT** (default

Bit	R/W	Reset	Description
11:9	R	0	Reserved
8	R	0h	**T_PCIE2_RP_VEND_XP_STATS0_CRC_ERRORS_LC**: Loads the current value from the CRC error counter into the VEND_XP_STATS1 register, and then clears the CRC error counter. The counter begins counting again immediately after it is cleared. Write a 1 to this field to trigger the load-and-clear operation. 0h **CRC_ERRORS_LC_INIT** (default)
7:5	R	0	Reserved
4	R/W	0h	**T_PCIE2_RP_VEND_XP_STATS0_8B10B_ERRORS_INF**: When set to 1, the 8b/10b error counter accumulates indefinitely (i.e. the one microsecond timer is ignored). 0h **8B10B_ERRORS_INF_INIT** (default)
3:1	R	0	Reserved
0	R	0h	**T_PCIE2_RP_VEND_XP_STATS0_8B10B_ERRORS_LC**: Loads the current value from the 8b/10b error counter into the VEND_XP_STATS1 register, and then clears the 8b/10b error counter. The counter begins counting again immediately after it is cleared. Write a 1 to this field to trigger the load-and-clear operation. 0h **8B10B_ERRORS_LC_INIT** (default)

32.2.7.33 T_PCIE2_RP_VEND_XP_STATS1 (0xF5Ch)

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: F5Ch | Read/Write: R

Bit	R/W	Reset	Description
31:24	R	None	**T_PCIE2_RP_VEND_XP_STATS1_8B10B_ERRORS**: Counts the number of 8b/10b errors detected in one microsecond. Saturates at 255 (counter does not roll over to 0).
23:16	R	None	**T_PCIE2_RP_VEND_XP_STATS1_CRC_ERRORS**: Counts the number of CRC errors detected in 64 microseconds. Saturates at 255 (counter does not roll over to 0).
15:8	R	None	**T_PCIE2_RP_VEND_XP_STATS1_NAKS_RCVD**: Counts the number of NAKs received from the endpoint in 64 microseconds. Saturates at 255 (counter does not roll over to 0).
7:0	R	None	**T_PCIE2_RP_VEND_XP_STATS1_FAILED_L0S_EXITS**: Counts the number of times the RX side of the XP went into Recovery in one millisecond, due to a failed attempt to exit L0s. Saturates at 255 (counter does not roll over to 0).

32.2.7.34 T_PCIE2_RP_VEND_ERROR_COUNT (0xF60h)

This register contains 8-bit saturating counters for some of RXL and TXBA reported errors.

Offset: F60h | Read/Write: R

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R	None	**T_PCIE2_RP_VEND_ERROR_COUNT_REPLAY**: Counts number of times TXBA replays. Saturates at 8'hFF. Can be cleared by writing 1 to PRIV_ERRSTS_REPLAY_STARTED_ERR.

Bit	R/W	Reset	Description
15:8	R	None	**T_PCIE2_RP_VEND_ERROR_COUNT_BAD_TLP**: Counts bad tlps reported by RXL (sum of lcrcl_err and bad_seq). Saturates at 8'hFF. Can be cleared by writing 1 to ERPTCAP_CERR_BAD_TLP
7:0	R	None	**T_PCIE2_RP_VEND_ERROR_COUNT_LCRC_ERR**: Counts LCRCL Errors reported by RXL. Saturates at 8'hFF. Can be cleared by writing 1 to PRIV_ERRSTS_LCRC_ERR.

32.2.7.35 T_PCIE2_RP_CFG_MISC (0xF64h)

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: F64h | Read/Write: R/W

Bit	R/W	Reset	Description
31:8	R	0	Reserved
7:0	R/W	0h	**T_PCIE2_RP_CFG_MISC_MUTE_IDLE**: MUTE mode is a power management feature in which we can gate cpu_clk and m2clk, if all the units are idle. A particular unit is idle when it has no outstanding transactions to be sent out. The MUTE_IDLE field is used to delay the idle assertion from xvr to clk. It should be programmed to cover the synchronization delay from cpuclk to m2clk. 0h **MUTE_IDLE_MIN** (default) FFh **MUTE_IDLE_MAX**

32.2.7.36 T_PCIE2_RP_PRIV_XP_INIT_RECOVERY (0xF68h)

Each new time frame, the number of 8b10b errors is counted. As soon as the number of 8b10b errors is equal to the threshold, the link is sent into recovery. If at the end of the time frame, the number of 8b10b errors is smaller than the threshold, a new time frame is started in which the 8b10b errors are counted.

Offset: F68h | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	0h	**T_PCIE2_RP_PRIV_XP_INIT_RECOVERY_8B10B_ERROR_ENABLE**: This bit enables the feature T_PCIE2_RP_ERRORRATE_8B10B_ENABLE_OFF disables the feature T_PCIE2_RP_ERRORRATE_8B10B_ENABLE_ON enables the feature 0h **8B10B_ERROR_ENABLE_INIT** (default)
30:20	R/W	64h	**T_PCIE2_RP_PRIV_XP_INIT_RECOVERY_8B10B_ERROR_WINDOW**: Programs the timeframe in multiples of 1 microsecond. 31 23 22 1 0 T_PCIE2_RP_PRIV_XP_INIT_RECOVERY_8B10B 64h **8B10B_ERROR_WINDOW_100US** (default)
19:0	R/W	9C4h	**T_PCIE2_RP_PRIV_XP_INIT_RECOVERY_8B10B_ERROR_THRESHOLD**: Bit [21:1] programs the 8b10b error threshold. If the number of errors in the programmed timeframe is equal to the threshold, the link is sent into recovery. 9C4h **8B10B_ERROR_THRESHOLD_2500** (default)

32.2.7.37 T_PCIE2_RP_PRIV_XP_LCTRL_2 (0xF6Ch)

Offset: F6Ch | Read/Write: R/W

Bit	R/W	Reset	Description
31	R	None	**T_PCIE2_RP_PRIV_XP_LCTRL_2_UPCONFIGURE_CAPABLE**: Gen2 link width up-configure capability of remote device as recorded from data rate id of received TS ordered sets.
30	R/W	0h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_REV2P0_COMPLIANCE_DIS**: To disable advertising rev2.0 support and link width up-configure capability. 0h **REV2P0_COMPLIANCE_DIS_INIT** (default)
29	R/W	1h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_EIDLE_INFERENCE_EN**: To disable electrical idle inference. It is enabled by default. 1h **EIDLE_INFERENCE_EN_INIT** (default)
28	R/W	1h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_SURPRISE_IDLE_USE_STAT_IDLE**: When set, use idle status signals from the pad to detect surprise electrical idle entry when running in Gen1 speed. 1h **SURPRISE_IDLE_USE_STAT_IDLE_INIT** (default)
27	R/W	1h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_ALLOW_SPEED_CHANGE_FROM_L1**: When set to 1, allow speed change to be initiated from the L1 or Rx_L0s link states. 1h **ALLOW_SPEED_CHANGE_FROM_L1_INIT** (default)
26:24	R/W	0h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_RECOVERY_SPEED_TIMEOUT_ADJ**: To adjust the minimum length of time the transmitter stays in electrical idle in the Recover.Sped state. The value denotes the time in microseconds added. 0h **RECOVERY_SPEED_TIMEOUT_ADJ_INIT** (default)
23:20	R/W	6h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_N_EIE_SYMBOLS**: This represents the number of K28.7 symbols transmitted prior to transmitting the first FTS ordered set in the Tx.L0s.FTS state when running in Gen2 speed. 6h **N_EIE_SYMBOLS_INIT** (default)
19	R/W	0h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_DEEMPHASIS_STRAP**: This is the backdoor register for BIOS to write an initial value to the HWInit register PCIE2_RP_LINK_CONTROL_STATUS_2_SELECTABLE_DEEMPHASIS. 0 = -6db 1 = -3.5db 0h **DEEMPHASIS_STRAP_INIT** (default)
18	R/W	0h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_ENFORCE_DEEMPHASIS**: Forces root port to use de-emphasis value specific in Link Control 2 Selectable De-emphasis field instead of the value requested by the endpoint (advertised in TSs). This bit is reserved for endpoint. Default value is 0b. Functionality of this bit is no longer needed since TX_PEAK_R2_SEL1/SEL0 registers provide complete control over AMP/PEAK/PEAK_PRE settings in gen2. NOTE: this field is reset by Cold Reset 0h **ENFORCE_DEEMPHASIS_INIT** (default)
17	R/W	0h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_POLLING_PREDETERMINED_LANES**: When set to 1, only lane 0 or 15 needs to detect an exit from electrical idle before moving from Polling.Active to Polling.Config. This bit is not yet implemented. 0h **POLLING_PREDETERMINED_LANES_DISABLE** (default)
16	R/W	0h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_AUTONOMOUS_CHANGE**: When set to 1, it indicates that the Upstream port-initiated link width/speed change is not caused by a link reliability issue. 0h **AUTONOMOUS_CHANGE_INIT** (default)

Bit	R/W	Reset	Description
15:12	R	1h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_DATA_RATE_SUPPORTED_REMOTE**: Gen2 data rate supported by other side as recorded from data rate identifier of received TS ordered sets. 1h **DATA_RATE_SUPPORTED_REMOTE_2P5** (default) 2h **DATA_RATE_SUPPORTED_REMOTE_5P0_2P5**
11:8	R/W	2h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_DATA_RATE_SUPPORTED**: When set to 4'b01, the supported link speed reported in the Link Capabilities Register is 2.5Gb/s. When set to 4'b10, the supported link speeds reported in the Link Capabilities Register are 5.0Gb/s and 2.5Gb/s. 1h **DATA_RATE_SUPPORTED_2P5** 2h **DATA_RATE_SUPPORTED_5P0_2P5** (default)
7:4	R/W	2h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_TARGET_LINK_SPEED**: Specifies the link rate to change to. This is reflected in the data rate advertised in the data rate identifier field of TS ordered sets. 1h **TARGET_LINK_SPEED_2P5** 2h **TARGET_LINK_SPEED_5P0** (default)
3	R/W	0h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_CTL_TX_MARGIN_OVERRIDE**: 0h **CTL_TX_MARGIN_OVERRIDE_DISABLED** (default) 1h **CTL_TX_MARGIN_OVERRIDE_ENABLED**
2	R	0	Reserved
1	R/W	0h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_ADVERTISED_RATE_CHANGE**: A write of 1 to this bit triggers the LTSSM to enter Recovery state, without setting the speed change bit, to change the advertised rate. This bit returns a value of 0 on a read. 0h **ADVERTISED_RATE_CHANGE_ZERO** (default)
0	R/W	0h	**T_PCIE2_RP_PRIV_XP_LCTRL_2_SPEED_CHANGE**: A write of 1 to this bit triggers the link speed negotiation procedure. This bit returns a value of 0 on a read. 0h **SPEED_CHANGE_ZERO** (default)

32.2.7.38 T_PCIE2_RP_PRIV_XP_PAD_PWRUP (0xF74h)

Offset: F74h | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R	0	Reserved
23:16	R/W	0h	**T_PCIE2_RP_PRIV_XP_PAD_PWRUP_PMRX_PWRUP_THRESHOLD**: It is the time to hold CDR at reset in Recovery.RcvrLock while waiting for power rails to be stable 0h **PMRX_PWRUP_THRESHOLD_INIT** (default)
15:0	R	0	Reserved

32.2.7.39 T_PCIE2_RP_PRIV_XP_RECOVERY_REASONS (0xF84h)

This register records the reasons we went into recovery. Each bit represents a different reason. Search for "recovery_reason" in the linkfsm to see what each bit means, as it's bound to change project to project. Writing a 0 clears the register and puts us into "record ALL recoveries encountered" mode. Writing a 1 clears the register and puts us into "record NEXT recovery encountered" mode. Default mode is "record ALL".

Offset: F84h | Read/Write: R/W

Bit	R/W	Reset	Description
31:0	R/W	0h	**T_PCIE2_RP_PRIV_XP_RECOVERY_REASONS_VALUE**: 0h **VALUE_INIT** (default) 0h **VALUE_REC_ALL** 1h **VALUE_REC_NEXT**

32.2.7.40 T_PCIE2_RP_PRIV_XP_RECOVERY_COUNT (0xF88h)

This register reflects the number of RECOVERY STATE entries in XP by the XVR. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: F88h | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	**T_PCIE2_RP_PRIV_XP_RECOVERY_COUNT_VALUE**: 0h **VALUE_INIT** (default)

32.2.7.41 T_PCIE2_RP_PRIV_XP_RX_L0S_ENTRY_COUNT (0xF8Ch)

This register reflects the number of entries from L0 to L0s at ltssm RX side. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: F8Ch | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	**T_PCIE2_RP_PRIV_XP_RX_L0S_ENTRY_COUNT_VALUE**: 0h **VALUE_INIT** (default)

32.2.7.42 T_PCIE2_RP_PRIV_XP_TX_L0S_ENTRY_COUNT (0xF90h)

This register reflects the number of entries from L0 to L0s at ltssm TX side. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: F90h | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	**T_PCIE2_RP_PRIV_XP_TX_L0S_ENTRY_COUNT_VALUE**: 0h **VALUE_INIT** (default)

32.2.7.43 T_PCIE2_RP_PRIV_XP_L1_ENTRY_COUNT (0xF94h)

This register reflects the number of entries from L0 to L1 It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read

Offset: F94h | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	**T_PCIE2_RP_PRIV_XP_L1_ENTRY_COUNT_VALUE**: 0h **VALUE_INIT** (default)

32.2.7.44 T_PCIE2_RP_PRIV_XP_L1_TO_RECOVERY_COUNT (0xF98h)

This register reflects the number of entries from L1 to Recovery. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: F98h | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	**T_PCIE2_RP_PRIV_XP_L1_TO_RECOVERY_COUNT_VALUE**: 0h **VALUE_INIT** (default)

32.2.7.45 T_PCIE2_RP_PRIV_XP_L0_TO_RECOVERY_COUNT (0xF9Ch)

This register reflects the number of entries from L0 to Recovery. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: F9Ch | Read/Write: R/W

Bit	R/W	Reset	Description
31:9	R	0	Reserved
8	R/W	0h	**T_PCIE2_RP_PRIV_XP_L0_TO_RECOVERY_COUNT_REASON**: 0h **REASON_INIT** (default) 0h **REASON_ALL** 1h **REASON_ERR**
7:0	R	0h	**T_PCIE2_RP_PRIV_XP_L0_TO_RECOVERY_COUNT_VALUE**: 0h **VALUE_INIT** (default)

32.2.7.46 T_PCIE2_RP_PRIV_XP_L1P_ENTRY_COUNT (0xFA0h)

This register reflects the number of entries from L0 to Deep L1. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: FA0h | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	**T_PCIE2_RP_PRIV_XP_L1P_ENTRY_COUNT_VALUE**: 0h **VALUE_INIT** (default)

32.2.7.47 T_PCIE2_RP_PRIV_XP_ASLM_COUNT (0xFA4h)

This register reflects the number of switching between x1 and x16 for Gen2 Only. It is reset whenever it is read. If the count saturates before it is read then it remains saturated till it is read.

Offset: FA4h | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	**T_PCIE2_RP_PRIV_XP_ASLM_COUNT_VALUE**: 0h **VALUE_INIT** (default)

32.2.7.48 T_PCIE2_RP_RP_VEND_CTL2 (0xFA8h)

This register contains miscellaneous control fields.

Offset: FA8h | Read/Write: R/W

Bit	R/W	Reset	Description
31:25	R	0h	Reserved
24	R/W	0h	**T_PCIE2_RP_VEND_CTL2_COMPLIANCE_X8_DELAY**: CTL for compliance delay pattern If set to 0, delay pattern will wrap as soon as the pattern hits lane n - 1 where n is the max width of the controller for the current xbar configuration. This ensures that there is always at least one lane sending a delay pattern during compliance. If set to 1, delay pattern will wrap when the pattern hits lane 8 or 16 (mod 8) even if it's max width is narrower than x8. This means there will be stretches when no lane is sending a delay pattern during compliance. 0h **COMPLIANCE_X8_DELAY_INIT** (default)
23	R/W	1h	**T_PCIE2_RP_VEND_CTL2_IGNORE_ATTENTION_BUTTON_MSG**: When this bit is set, DUT will ignore upstream Attention button pressed message which has been send by endpoint. PCIe spec 1.1 specifies that we should ignore this message. 0h **IGNORE_ATTENTION_BUTTON_MSG_FALSE** 1h **IGNORE_ATTENTION_BUTTON_MSG_TRUE** (default)
22	R/W	0h	**T_PCIE2_RP_VEND_CTL2_UNBLOCK_UP_TRANSACTIONS**: When an upstream error has occured which causes all further upstream transaaction to be blocked, writing one to this bit will clean blocking of all upstream transacton. 0h **UNBLOCK_UP_TRANSACTIONS_FALSE** (default) 1h **UNBLOCK_UP_TRANSACTIONS_TRUE**
21	R/W	0h	**T_PCIE2_RP_VEND_CTL2_BLOCK_UP_TRANSACTIONS_ON_ERR**: To block all upstream transaction after an Error has been detected and forwarding of packet after the error can cause data corruption or break programming paradigm. 0h **BLOCK_UP_TRANSACTIONS_ON_ERR_EN** (default) 1h **BLOCK_UP_TRANSACTIONS_ON_ERR_DIS**
20	R/W	0h	**T_PCIE2_RP_VEND_CTL2_HW_AUTO_WIDTH_DISABLE_DIS**: To hide the LINK_CONTROL_STATUS_HW_AUTO_WIDTH_DISABLE register. 1h **HW_AUTO_WIDTH_DISABLE_DIS_TRUE** 0h **HW_AUTO_WIDTH_DISABLE_DIS_FALSE** (default)
19	R/W	0h	**T_PCIE2_RP_VEND_CTL2_BW_MANAGEMENT_INT_EN_DIS**: To hide the LINK_CAPABILITIES_BW_MANAGEMENT_INT_EN register. 1h **BW_MANAGEMENT_INT_EN_DIS_TRUE** 0h **BW_MANAGEMENT_INT_EN_DIS_FALSE** (default)
18	R/W	0h	**T_PCIE2_RP_VEND_CTL2_AUTO_BANDWIDTH_INT_EN_DIS**: To hide the LINK_CAPABILITIES_AUTO_BANDWIDTH_INT_EN register. 1h **AUTO_BANDWIDTH_INT_EN_DIS_TRUE** 0h **AUTO_BANDWIDTH_INT_EN_DIS_FALSE** (default)
17	R/W	0h	**T_PCIE2_RP_VEND_CTL2_AUTO_BANDWIDTH_DIS**: To hide the LINK_CAPABILITIES_AUTO_BANDWIDTH register. 1h **AUTO_BANDWIDTH_DIS_TRUE** 0h **AUTO_BANDWIDTH_DIS_FALSE** (default)
16	R/W	0h	**T_PCIE2_RP_VEND_CTL2_BW_MANAGEMENT_DIS**: To hide the LINK_CAPABILITIES_BW_MANAGEMENT register. 1h **BW_MANAGEMENT_DIS_TRUE** 0h **BW_MANAGEMENT_DIS_FALSE** (default)
15:14	R	0	Reserved
13	R/W	1h	**T_PCIE2_RP_VEND_CTL2_SHADOW_LINK_BW_NOTIFY_CAP**: 1h **SHADOW_LINK_BW_NOTIFY_CAP_EN** (default) 0h **SHADOW_LINK_BW_NOTIFY_CAP_DIS**
12	R	0h	**T_PCIE2_RP_VEND_CTL2_GEN2_PROTOCOL_DISABLE**: 1h **ON** 0h **OFF** (default)

Bit	R/W	Reset	Description
11	R/W	1h	**T_PCIE2_RP_VEND_CTL2_SLOT_IMPLEMENTED**: 1h **SLOT_IMPLEMENTED_INIT** (default) 1h **SLOT_IMPLEMENTED_YES** 0h **SLOT_IMPLEMENTED_NO**
10	R	0	Reserved
9	R/W	0h	**T_PCIE2_RP_VEND_CTL2_ERR_STS_HOTPLUG_PME**: NOTE: this field is reset by Cold Reset 0h **ERR_STS_HOTPLUG_PME_FALSE** (default) 1h **ERR_STS_HOTPLUG_PME_TRUE** 1h **ERR_STS_HOTPLUG_PME_CLEAR**
8	R/W	0h	**T_PCIE2_RP_VEND_CTL2_ERR_STS_HOTPLUG_NMI**: NOTE: this field is reset by Cold Reset 0h **ERR_STS_HOTPLUG_NMI_FALSE** (default) 1h **ERR_STS_HOTPLUG_NMI_TRUE** 1h **ERR_STS_HOTPLUG_NMI_CLEAR**
7	R/W	1h	**T_PCIE2_RP_VEND_CTL2_PCA_ENABLE**: This bit will enable/disable the PCA in any TMS. CTLR0's bit controls the enable/disable in a TMS. Default value is 1'b1(ENABLE_ON). 1h **PCA_ENABLE_ON** (default) 0h **PCA_ENABLE_OFF**
6	R	0	Reserved
5	R/W	0h	**T_PCIE2_RP_VEND_CTL2_GEN2_SPEED_DISABLE**: This bit will limit the max link speed to gen1 when set. 1h **GEN2_SPEED_DISABLE_ON** 0h **GEN2_SPEED_DISABLE_OFF** (default)
4	R/W	0h	**T_PCIE2_RP_VEND_CTL2_GEN2_PROTOCOL_DISABLE**: This will disable gen2 protocol and will force the use of Gen 1.1 protocol. 0h **GEN2_PROTOCOL_DISABLE_OFF** (default) 1h **GEN2_PROTOCOL_DISABLE_ON**
3	R	0	Reserved
2	R/W	1h	**T_PCIE2_RP_VEND_CTL2_DEV_CAP_EXTENDED_TAG_FIELD_SIZE**: This is shadow field of the T_PCIE2_RP_DEVICE_CAPABILITY_EXTENDED_TAG_FIELD_SIZE. Values written to this field will be reflected in EXTENDED_TAG_FIELD_SIZE of T_PCIE2_RP_DEVICE_CAPABILITY. 1h **DEV_CAP_EXTENDED_TAG_FIELD_SIZE_8B** (default) 0h **DEV_CAP_EXTENDED_TAG_FIELD_SIZE_5B**
1	R/W	0h	**T_PCIE2_RP_VEND_CTL2_DIS_MA_TA_EP_MERGE**: When set disables the logic which takes care of merging of EP/MA/TA when mergin is enabled on CPL pkt (downstream). 1h **DIS_MA_TA_EP_MERGE_ON** 0h **DIS_MA_TA_EP_MERGE_OFF** (default)
0	R	0	Reserved

32.2.7.49 T_PCIE2_RP_PRIV_XP_CONFIG (0xFACh)

Offset: FACh | Read/Write: R/W

Bit	R/W	Reset	Description
31:2	R	0	Reserved

Bit	R/W	Reset	Description
1:0	R/W	0h	**T_PCIE2_RP_PRIV_XP_CONFIG_LOW_PWR_DURATION**: Meant for selection of different types of the information logging for the Health and Performance counters. Primarliy this register added to support the selection of different information for the low power duration count. Used for selecting the Low power information to be presented in the T_PCIE2_RP_PRIV_XP_DURATION_IN_LOW_PWR_100NS. TX_L0S, RX_L0S and L1 collect the duration for which LTSSM was in that state. DURATION_IDLE selection gives following IDLE = Duration(TX_L0S && RX_L0S) + Duration of L1 0h **LOW_PWR_DURATION_TX_L0S** (default) 1h **LOW_PWR_DURATION_RX_L0S** 2h **LOW_PWR_DURATION_L1** 3h **LOW_PWR_DURATION_IDLE**

32.2.7.50 T_PCIE2_RP_PRIV_XP_DURATION_IN_LOW_PWR_100NS (0xFB0h)

This register counts the duration in multiples of 100ns for the type of low power information selected by the T_PCIE2_RP_PRIV_XP_CONFIG_LOW_PWR_DURATION.

Offset: FB0h | Read/Write: R

Bit	R/W	Reset	Description
31:0	R	0h	**T_PCIE2_RP_PRIV_XP_DURATION_IN_LOW_PWR_100NS_VALUE**: 0h **VALUE_INIT** (default)

32.2.7.51 T_PCIE2_RP_PRIV_XP_SKP_TIMEOUT (0xFB4h)

This register counts the duration in multiples of 100ns for the type of low power information selected by the T_PCIE2_RP_PRIV_XP_CONFIG_LOW_PWR_DURATION.

Offset: FB4h | Read/Write: R/W

Bit	R/W	Reset	Description
31:26	R	0	Reserved
25:13	R/W	569h	**T_PCIE2_RP_PRIV_XP_SKP_TIMEOUT_THRESHOLD_GEN2**: 569h **THRESHOLD_GEN2_INIT** (default) 271h **THRESHOLD_GEN2_250** 2B4h **THRESHOLD_GEN2_278** 30Dh **THRESHOLD_GEN2_313** 37Ah **THRESHOLD_GEN2_357** 410h **THRESHOLD_GEN2_417** 4DFh **THRESHOLD_GEN2_500** 569h **THRESHOLD_GEN2_555**
12:0	R/W	569h	**T_PCIE2_RP_PRIV_XP_SKP_TIMEOUT_THRESHOLD**: 569h **THRESHOLD_INIT** (default) 4E2h **THRESHOLD_250** 569h **THRESHOLD_278** 61Ah **THRESHOLD_313** 6F5h **THRESHOLD_357** 821h **THRESHOLD_417** 9BFh **THRESHOLD_500** AD2h **THRESHOLD_555**

32.2.7.52 T_PCIE2_RP_PRIV_XP_EIDLE_INFERENCE_TIMEOUT (0xFB8h)

Offset: FB8h | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	6F2h	**T_PCIE2_RP_PRIV_XP_EIDLE_INFERENCE_TIMEOUT_B**: This sets the UI window for electrical idle exit not detected in the Recover.Speed state on an unsuccessful speed negotiation. 6F2h **B_INIT** (default)
15:0	R/W	8Fh	**T_PCIE2_RP_PRIV_XP_EIDLE_INFERENCE_TIMEOUT_A**: This sets the UI window for COM not detected in the Recovery.Speed state on a successful speed negotiation. 8Fh **A_INIT** (default)

32.2.7.53 T_PCIE2_RP_VEND_SHADOW (0xFC8h)

This Register is shadow register of T_PCIE2_RP_SS_1. The register must be programmed at boot time with the SubSystemVendorID and a unique non-zero number assigned by that vendor for this product. Register T_PCIE2_RP_SS_1 must be read-only at runtime in config space.

Offset: FC8h | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R/W	0h	**T_PCIE2_RP_VEND_SHADOW_SS_1_SSID**: 0h **SSID_INIT** (default)
15:0	R/W	10DEh	**T_PCIE2_RP_VEND_SHADOW_SS_1_SSVID**: 10DEh **SSVID_INIT** (default)

32.2.7.54 T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP (0xFCCh)

These registers contain the mapping of the Transmit Margin field in the Link Control 2 register to the TX_AMP signal which will be driven onto the pads.

Offset: FCCh | Read/Write: R/W

Bit	R/W	Reset	Description
31:30	R/W	1h	**T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP_CTL**: 1h **CTL_INIT** (default)
29:24	R/W	0h	**T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP_CODE3**: 0h **CODE3_INIT** (default) 14h **CODE3_DESKTOP** 8h **CODE3_MOBILE**
23:22	R	0	Reserved
21:16	R/W	0h	**T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP_CODE2**: 0h **CODE2_INIT** (default) 18h **CODE2_DESKTOP** Ah **CODE2_MOBILE**
15:14	R	0	Reserved
13:8	R/W	0h	**T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP_CODE1**: 0h **CODE1_INIT** (default) 1Ch **CODE1_DESKTOP** Ch **CODE1_MOBILE**
7:6	R	0	Reserved

Bit	R/W	Reset	Description
5:0	R/W	0h	**T_PCIE2_RP_TX_MARGIN_MAP0_TX_AMP_CODE0**: 0h **CODE0_INIT** (default) 20h **CODE0_DESKTOP** Ch **CODE0_MOBILE**

32.2.7.55 T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP (0xFD0h)

Offset: FD0h | Read/Write: R/W

Bit	R/W	Reset	Description
31:30	R/W	1h	**T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP_CTL**: 1h **CTL_INIT** (default)
29:24	R/W	0h	**T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP_CODE7**: 0h **CODE7_INIT** (default) 4h **CODE7_DESKTOP** 0h **CODE7_MOBILE**
23:22	R	0	Reserved
21:16	R/W	0h	**T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP_CODE6**: 0h **CODE6_INIT** (default) 8h **CODE6_DESKTOP** 2h **CODE6_MOBILE**
15:14	R	0	Reserved
13:8	R/W	0h	**T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP_CODE5**: 0h **CODE5_INIT** (default) Ch **CODE5_DESKTOP** 4h **CODE5_MOBILE |**
7:6	R	0	Reserved
5:0	R/W	0h	**T_PCIE2_RP_TX_MARGIN_MAP1_TX_AMP_CODE4**: 0h **CODE4_INIT** (default) 10h **CODE4_DESKTOP** 6h **CODE4_MOBILE**

32.2.7.56 T_PCIE2_RP_CTL_2 (0xFD4h)

Offset: FD4h | Read/Write: R/W

Bit	R/W	Reset	Description
31	R	0	Reserved
30:28	R/W	0h	**T_PCIE2_RP_CTL_2_RX_EQ_R1_1C**: 0h **RX_EQ_R1_1C_INIT** (default) 0h **RX_EQ_R1_1C_NO_EQ** 7h **RX_EQ_R1_1C_MAX_EQ**
27:20	R	0	Reserved
19:17	R/W	0h	**T_PCIE2_RP_CTL_2_TX_PEAK_PRE_R1_1C**: 0h **TX_PEAK_PRE_R1_1C_INIT** (default)

Bit	R/W	Reset	Description
16:12	R	Eh	**T_PCIE2_RP_CTL_2_TX_PEAK_R1_1C**: Eh **TX_PEAK_R1_1C_INIT** (default) 0h **TX_PEAK_R1_1C_DISABLE** Eh **TX_PEAK_R1_1C_37DB** 14h **TX_PEAK_R1_1C_60DB**
11:8	R/W	0h	**T_PCIE2_RP_CTL_2_TX_CMADJ_R1_1C**: 0h **TX_CMADJ_R1_1C_INIT** (default)
7:6	R	0	Reserved
5:0	R/W	20h	**T_PCIE2_RP_CTL_2_TX_AMP_R1_1C**: 20h **TX_AMP_R1_1C_INIT** (default) 26h **TX_AMP_R1_1C_1150_MVPPD** 20h **TX_AMP_R1_1C_1000_MVPPD** Ch **TX_AMP_R1_1C__500_MVPPD** 0h **TX_AMP_R1_1C__200_MVPPD**

32.2.7.57 T_PCIE2_RP_CTL_3 (0xFD8h)

Offset: FD8h | Read/Write: R/W

Bit	R/W	Reset	Description
31	R	0	Reserved
30:28	R/W	0h	**T_PCIE2_RP_CTL_3_RX_EQ_R2_1C**: 0h **RX_EQ_R2_1C_INIT** (default) 0h **RX_EQ_R2_1C_NO_EQ** 7h **RX_EQ_R2_1C_MAX_EQ**
27:25	R/W	0h	**T_PCIE2_RP_CTL_3_TX_PEAK_PRE_R2_SEL0_1C**: 0h **TX_PEAK_PRE_R2_SEL0_1C_INIT** (default)
24:20	R/W	14h	**T_PCIE2_RP_CTL_3_TX_PEAK_R2_SEL0_1C**: 14h **TX_PEAK_R2_SEL0_1C_INIT** (default) 0h **TX_PEAK_R2_SEL0_1C_DISABLE** Eh **TX_PEAK_R2_SEL0_1C_37DB** 14h **TX_PEAK_R2_SEL0_1C_60DB**
19:17	R/W	0h	**T_PCIE2_RP_CTL_3_TX_PEAK_PRE_R2_SEL1_1C**: 0h **TX_PEAK_PRE_R2_SEL1_1C_INIT** (default)
16:12	R/W	Eh	**T_PCIE2_RP_CTL_3_TX_PEAK_R2_SEL1_1C**: Eh **TX_PEAK_R2_SEL1_1C_INIT** (default) 0h **TX_PEAK_R2_SEL1_1C_DISABLE** Eh **TX_PEAK_R2_SEL1_1C_37DB** 14h **TX_PEAK_R2_SEL1_1C_60DB**
11:8	R/W	0h	**T_PCIE2_RP_CTL_3_TX_CMADJ_R2_1C**: 0h **TX_CMADJ_R2_1C_INIT** (default)
7:6	R	0	Reserved
5:0	R/W	20h	**T_PCIE2_RP_CTL_3_TX_AMP_R2_1C**: 20h **TX_AMP_R2_1C_INIT** (default) 26h **TX_AMP_R2_1C_1150_MVPPD** 20h **TX_AMP_R2_1C_1000_MVPPD** Ch **TX_AMP_R2_1C__500_MVPPD** 0h **TX_AMP_R2_1C__200_MVPPD**

32.2.7.58 T_PCIE2_RP_TIMEOUT2 (0xFDCh)

Offset: FDCh | Read/Write: R/W

Bit	R/W	Reset	Description
31:5	R	0	Reserved
4:0	R/W	Ah	**T_PCIE2_RP_TIMEOUT2_MIN_L1_L2_IDLE_TIME**: Sets minimum amount of time we stay in L1/L2. Holds off exit-condition detection for this amount of time (to prevent accidental wake from data before E-Idle (e.g. 2nd EIOS in Gen2) Measured in units of 4ns scaled with XCLK frequency Ah **MIN_L1_L2_IDLE_TIME_INIT** (default)

32.2.7.59 T_PCIE2_RP_PRIV_MISC (0xFE0h)

Offset: FE0h | Read/Write: R/W

Bit	R/W	Reset	Description
31	R/W	0h	**T_PCIE2_RP_PRIV_MISC_TMS_CLK_CLAMP_ENABLE**: Enables per-TMS XCLK/XTXCLK clamping using value from T_PCIE2_RP_PRIV_MISC_CLK_CLAMP_THRESHOLD to determine when to clamp. This will clamp all the logic inside a TMS shared between controllers. Note: This enable is per TMS. Use this TMS's controller 0's register to set this. 0h **TMS_CLK_CLAMP_ENABLE_INIT** (default)
30:24	R/W	3h	**T_PCIE2_RP_PRIV_MISC_TMS_CLK_CLAMP_THRESHOLD**: Number of contiguous nbref_clks (25MHz) after clamping requirements are met (L1, lanes powered down if necessary, tx FIFO empty, no eidle exit, etc) before actually clamping xclk/xtxclk. Choose a value large enough to allow all upstream transactions to clear the XP (make it into the fpci clk domain) and for the sleep signals coming from the ltssm to make it to the pads. If the value is too small, upstream transactions will get stuck in the XP and/or the sleep signals will not assert properly. The value must also be large enough so that config transactions have time to propogate from pri domain to xclk domain. Note: This enable is per TMS. Use this TMS's controller 0's register to set this. 3h **TMS_CLK_CLAMP_THRESHOLD_INIT** (default)
23	R/W	0h	**T_PCIE2_RP_PRIV_MISC_CTLR_CLK_CLAMP_ENABLE**: Enables per-controller XCLK/XTXCLK clamping using value from T_PCIE2_RP_PRIV_MISC_CTLR CLK_CLAMP_THRESHOLD to determine when to clamp. 0h **CTLR_CLK_CLAMP_ENABLE_INIT** (default)
22:16	R/W	1h	**T_PCIE2_RP_PRIV_MISC_CTLR_CLK_CLAMP_THRESHOLD**: Number of contiguous nbref_clks (25MHz) after clamping requirements are met (L1, lanes powered down if necessary, tx FIFO empty, no eidle exit, etc) before actually clamping xclk/xtxclk. Choose a value large enough to for the sleep signals coming from the ltssm to make it to the pads. If the value is too small,the sleep signals will not assert properly. 1h **CTLR_CLK_CLAMP_THRESHOLD_INIT** (default)
15:4	R	0	Reserved
3:0	R/W	Fh	**T_PCIE2_RP_PRIV_MISC_PRSNT_MAP**: Fh **PRSNT_MAP_INIT** (default)

32.2.7.60 T_PCIE2_RP_VEND_XP3 (0xFE8h)

This register is the Symbol Alignment Error Handling register.

Offset: FE8h | Read/Write: R/W

Bit	R/W	Reset	Description
31:8	R	0	Reserved

Bit	R/W	Reset	Description
7:0	R/W	3h	**T_PCIE2_RP_VEND_XP3_SA_ERROR_LIMIT**: Specifies number of misaligned COM symbols we need to see before declaring loss of Symbol Alignment. When Symbol Alignment is lost, the error bit for all subsequent symbols is asserted until 1 good COM symbol is seen. Then the error bit is deasserted and the counter tracking misaligned COM symbols is reset. Set to 0 to disable this feature. Note: Since 8b10b and symbol alignment errors are indistinguishable, symbol alignment errors (when error limit is hit) will contribute to all 8b10b error counts including the Goto Recovery on 8b10b errors feature. This is useful to speed up link retraining after losing symbol alignment. 3h **SA_ERROR_LIMIT_INIT** (default)

32.2.7.61 T_PCIE2_RP_XP_CTL_1 (0xFECh)

This register contains control registers used in XP.

Offset: FECh | Read/Write: R/W

Bit	R/W	Reset	Description
31	R	1h	Reserved
30:28	R/W	1h	**T_PCIE2_RP_XP_CTL_1_SPARE**: 1h **SPARE_INIT** (default)
27	R/W	1h	**T_PCIE2_RP_XP_CTL_1_EXIT_L1_AT_CLKREQ_ASSERTION**: When set, ltssm will exit L1 when clkreq assertion happens. NOTE : This feature only applies to downstream port 0h **EXIT_L1_AT_CLKREQ_ASSERTION_DISABLED** 1h **EXIT_L1_AT_CLKREQ_ASSERTION_ENABLED** (default)
26	R/W	1h	**T_PCIE2_RP_XP_CTL_1_NEW_IOBIST_CTRL**: This bit when set will enable control signals from iobist to take effect. Some of the control signal will be are enabling/disabling scrambling, enabling and disabling cheking for SKP symbols. 0h **NEW_IOBIST_CTRL_DISABLED** 1h **NEW_IOBIST_CTRL_ENABLED** (default)
25	R/W	0h	**T_PCIE2_RP_XP_CTL_1_OLD_IOBIST_EN**: This bit when set will enable old iobist, else it will enable new iobist. 0h **OLD_IOBIST_EN_INIT** (default)
24:19	R/W	0h	**T_PCIE2_RP_XP_CTL_1_EIOS_DEBOUNCE_TIMER**: This is number of xclk for which rx_data_en signals need to be de-asserted before asserting it again. 0h **EIOS_DEBOUNCE_TIMER_INIT** (default)
18	R/W	0h	**T_PCIE2_RP_XP_CTL_1_DISABLE_RX_LANE_AFTER_EIOS**: Disable rx_lane_en after EIOS is seen in any of the enabled lane during recovery.Rlock and Recovery.Config in order to avoid coupling. 0h **DISABLE_RX_LANE_AFTER_EIOS_INIT** (default)
17	R/W	0h	**T_PCIE2_RP_XP_CTL_1_ENABLE_DESKEW_FOR_SPDCH_NEGOTIATION**: When set to 0, forces LTSSM to disable deskew in Recovery.RcvrCfg state if the transition from Recovery.RcvrLock to RcvrCfg state is due to LTSSM detecting eight consecutive training set on any of the lanes but not all of the lanes, and speed change bit is set to 1. When set to 1, LTSSM will always enable deskew in Recovery.RcvrCfg state. 0h **ENABLE_DESKEW_FOR_SPDCH_NEGOTIATION_INIT** (default)
16	R/W	0h	**T_PCIE2_RP_XP_CTL_1_BYPASS_CONFIG_PWRUP**: When set to 0, forces LTSSM to go from Recovery.RcvrLock to Config.Pwrup if LTSSM sees training set on some but not all the active lanes. This bit only takes effect when link is operating in dynamically reduced link width and spec-defined link width resize is enabled. When set to 1, LTSSM will go from Recovery.RcvrLock to Config.LinkStart directly, bypassing Config.Pwrup, when LTSSM sees training set on some but not all the active lanes. 0h **BYPASS_CONFIG_PWRUP_INIT** (default)

Bit	R/W	Reset	Description
15	R/W	0h	**T_PCIE2_RP_XP_CTL_1_RESET_TS_CTRL_ON_ERROR**: When set to 1, in Config.LinkStart state, LTSSM will reset the internal TS_CTRL0 register if 8b10b error is present on any symbol of the training set. When set to 0, in Config.LinkStart state, LTSSM will reset the internal TS_CTRL0 register if 8b10b error is present on symbols 0, 4 or 5 only. 0h **RESET_TS_CTRL_ON_ERROR_INIT** (default)
14	R/W	0h	**T_PCIE2_RP_XP_CTL_1_RX_EIDLE_EXIT_TIMER_IN_CONFIG**: When set to 1, T_PCIE2_RP_XP_IDLE_TIMER_1_RX_EIDLE_EXIT will be applied in Config.LinkStart only. When set to 0, T_PCIE2_RP_XP_IDLE_TIMER_1_RX_EIDLE_EXIT will be applied in Config.LinkStart, Recovery and Polling. 0h **RX_EIDLE_EXIT_TIMER_IN_CONFIG_INIT** (default)
13	R/W	1h	**T_PCIE2_RP_XP_CTL_1_FORCE_RESET_IN_CONFIG**: When set to 1, LTSSM will force receiver logic to be reset when enabling lanes in Config.LinkStart state for PCIe 2.0 spec-defined link width upconfiguration. 1h **FORCE_RESET_IN_CONFIG_INIT** (default)
12	R/W	1h	**T_PCIE2_RP_XP_CTL_1_LINK_RESIZE_PWRDN_CTL**: 1h **LINK_RESIZE_PWRDN_CTL_INIT** (default)
11	R/W	1h	**T_PCIE2_RP_XP_CTL_1_ALLOW_SPEED_CHANGE_FROM_L0S**: When set to 1, LTSSM will initiate link speed/width/advertised rate change while RX side is in L0s link state. The default value of this register is 1. 1h **ALLOW_SPEED_CHANGE_FROM_L0S_INIT** (default)
10	R/W	0h	**T_PCIE2_RP_XP_CTL_1_PRESERVE_TX_PACKET_ON_DL_RETRAIN**: When set to 1, LTSSM will wait for framer to be idle before entering Recovery for DL initiated link retrain. Note: Setting this bit to 1 could prevent LTSSM to enter Recovery on a replay rollover event if the retry buffer is very full. 0h **PRESERVE_TX_PACKET_ON_DL_RETRAIN_INIT** (default)
9	R/W	0h	**T_PCIE2_RP_XP_CTL_1_PRESERVE_TX_PACKET_ON_RECOVERY**: Note: Setting this bit to 1 does not work if L1 ASPM is enabled as it will cause hang during L1 negotiation if Rootport initiate entry to Recovery. Note: During DL init sequence, if link is retrained, setting this bit to 1 would hang the chip. 0h **PRESERVE_TX_PACKET_ON_RECOVERY_INIT** (default)
8	R/W	1h	**T_PCIE2_RP_XP_CTL_1_PRESERVE_TX_PACKET_ON_WIDTH_CHANGE**: When set to 1, LTSSM will wait for framer to be idle before entering Recovery for link width change. 1h **PRESERVE_TX_PACKET_ON_WIDTH_CHANGE_INIT** (default)
7	R/W	1h	**T_PCIE2_RP_XP_CTL_1_PRESERVE_TX_PACKET_ON_SPEED_CHANGE**: When set to 1, LTSSM will wait for framer to be idle before entering Recovery for link speed change or advertised rate change. 1h **PRESERVE_TX_PACKET_ON_SPEED_CHANGE_INIT** (default)
6	R/W	0h	**T_PCIE2_RP_XP_CTL_1_RESET_LANE_ENABLE_ORIG_IN_DETECT**: This fields will allow lane_enable_original to be reset in Detect state. 0h **RESET_LANE_ENABLE_ORIG_IN_DETECT_INIT** (default)
5:2	R/W	0h	**T_PCIE2_RP_XP_CTL_1_IDLE_TO_L0_DELAY**: This fields set the number of clocks that the LTSSM will delay the transition from Recovery.Idle to L0.Normal link state or Config.Idle to L0.Normal link state. 0h **IDLE_TO_L0_DELAY_INIT** (default)

Bit	R/W	Reset	Description
1	R/W	0h	**T_PCIE2_RP_XP_CTL_1_LWLO_HUNT_ON_BAD_TS1**: When set to 1, the LWLOFSM hunt timer does not reset when any lane receives an error on the TS1 ordered set in the Config.LinkStart, Config.LinkAccept, Config.LaneWait, and Config.LaneAccept states. When set to 0, the LWLOFSM hunt timer will reset when any lane receives the first error on the TS1 ordered set after any lanes has locked and hunt timer has started counting in the Config.LinkStart, Config.LinkAccept, Config.LaneWait, and Config.LaneAccept states. 0h **LWLO_HUNT_ON_BAD_TS1_INIT** (default)
0	R/W	0h	**T_PCIE2_RP_XP_CTL_1_FORCE_SA_IN_CONFIG**: When set to 1, enables symbol alignment in Config.LinkAccept, Config.LaneWait, Config.LaneAccept, and Config.Complete states. 0h **FORCE_SA_IN_CONFIG_INIT** (default)

32.2.7.62 T_PCIE2_RP_PRIV_XP_L1BEACON (0xFF0h)

Offset: FF0h | Read/Write: R/W

Bit	R/W	Reset	Description
31:10	R	0	Reserved
9:2	R/W	0h	**T_PCIE2_RP_PRIV_XP_L1BEACON_N_EIE_SYMBOLS**: It is number of EIE symbols sent in L1 Beacon Entry/Exist state 40h **N_EIE_SYMBOLS_INIT** (default)
1	R/W	0h	**T_PCIE2_RP_PRIV_XP_L1BEACON_TX_BYP_CTRL**: 0h **TX_BYP_CTRL_DEFAULT** (default) 0h **TX_BYP_CTRL_ONLYL0** 1h **TX_BYP_CTRL_ALLLANES**
0	R/W	0h	**T_PCIE2_RP_PRIV_XP_L1BEACON_EN**: 0h **EN_DEFAULT** (default)

32.2.7.63 T_PCIE2_RP_TIMEOUT3 (0xFF4h)

Offset: FF4h | Read/Write: R/W

Bit	R/W	Reset	Description
31:24	R/W	Ch	**T_PCIE2_RP_TIMEOUT3_RX_L0S_IDLE_TIME_GEN2**: This register controls the amount of time LTSSM delays the detection of electrical idle exit in the RX_L0s state at gen2 speed of operation. Ch **RX_L0S_IDLE_TIME_GEN2_INIT** (default)
23:16	R/W	4h	**T_PCIE2_RP_TIMEOUT3_RX_L0S_IDLE_TIME_GEN1**: This register controls the amount of time LTSSM delays the detection of electrical idle exit in the RX_L0s state at gen1 speed of operation. 4h **RX_L0S_IDLE_TIME_GEN1_INIT** (default)
15:8	R/W	FFh	**T_PCIE2_RP_TIMEOUT3_TX_L0S_IDLE_TIME**: This register controls the minimum amount of time LTSSM stays in TX_L0S_IDLE state Measured in units of 40ns scaled with XCLK frequency. A setting of 8'hFF disables the delay. FFh **TX_L0S_IDLE_TIME_INIT** (default)
7:4	R/W	0h	**T_PCIE2_RP_TIMEOUT3_RX_EIDLE_EXIT_DELAY**: This register controls the amount of time the LTSSM delays the detection of TS ordered set when exiting out of electrical idle state. Each increment represents 2^X microseconds. A setting of 0 disables the delay. 0h **RX_EIDLE_EXIT_DELAY_INIT** (default)

Bit	R/W	Reset	Description
3:0	R/W	0h	**T_PCIE2_RP_TIMEOUT3_TX_EIDLE_EXIT_DELAY**: This register controls the amount of time the LTSSM delays the transmission of TS ordered sets when exiting out of electrical idle state. Unit is in microseconds. Each increment represents 2^X microseconds. A setting of 0 disables the delay. 0h **TX_EIDLE_EXIT_DELAY_INIT** (default)

32.2.7.64 T_PCIE2_RP_XP_CTL_2 (0xFF8h)

This register contains controls bits used in XP.

LOOPBACK_EXIT_THRESHOLD: This register programs the maximum amount of time spent in Loopback.Exit state as loopback slave. Unit is in xclk. The setting of 0 disables the timer in Loopback.Exit state such that loopback slave relies on detection of electrical idle data to exit from Loopback.

Default is 64 xclks.

CONFIG_LINKSTART_REQUIRE_PAD_LANE_NUM: This register forces LTSSM training set detector to require PAD in the lane number symbol of training set in Config.LinkStart state.

Offset: FF8 | Read/Write: R/W

Bit	R/W	Reset	Description
31:9	R	0	Reserved
8	R/W	0h	**T_PCIE2_RP_XP_CTL_2_CONFIG_LINKSTART_REQUIRE_PAD_LANE_NUM**: 0h **INIT** (default)
7:0	R/W	40h	**T_PCIE2_RP_XP_CTL_2_LOOPBACK_EXIT_THRESHOLD**: 40h **INIT** (default)

32.2.7.65 T_PCIE2_RP_VEND_XP_STATS2 (0xFFCh)

This register contains fields unique to NVIDIA's implementation of PCI Express devices.

Offset: FFC | Read/Write: R/W

Bit	R/W	Reset	Description
31:16	R	0	Reserved
15:0	R/W	0h	**T_PCIE2_RP_VEND_XP_STATS2_8B10B_ERR_LANEMASK**: Each individual bit represent the corresponding logical lane. When set to 1, the 8b10b error from that specific lane will be masked out from the counter in T_PCIE2_RP_VEND_XP_STATS1_8B10B_ERRORS 0h **8B10B_ERR_LANEMASK_INIT** (default) FFFEh **8B10B_ERR_LANEMASK_LN0** FFFDh **8B10B_ERR_LANEMASK_LN1** FFFBh **8B10B_ERR_LANEMASK_LN2** FFF7h **8B10B_ERR_LANEMASK_LN3** FFEFh **8B10B_ERR_LANEMASK_LN4** FFDFh **8B10B_ERR_LANEMASK_LN5** FFBFh **8B10B_ERR_LANEMASK_LN6** FF7Fh **8B10B_ERR_LANEMASK_LN7**

32.3 PCIE Root Port Interface Registers

The registers mentioned here are registers internal to AXI to FPCI wrapper target. These registers will control both AXI initiator and target block behavior.

Register Name	Description
BARi mapping, i in [0,8]	Defines the size of BARi and where BARi is mapped in AXI and FPCI address space
MSI BAR	Defines the BAR used to detect MSI accesses from upstream FPCI
Configuration	Depends on production mode
Interrupt mask	Selectively drive interrupt to MPCORE
Interrupt type	Which interrupt has occurred
Interrupt signature	Identifies the last interrupt detected by AFI
AXI field overrides	Overrides for AXI initiator fields ACACHE/AUSER/APROT
PCIe 2.0 Config/Status I/O	Static configuration inputs to PCIe 2.0 and output status signals

32.3.1.1 AXI BARi Mapping

In the register name, i ranges from 0 to 8.

Field	Bits	R/W	Reset	Description
Size	19:0	R/W	See Note 1	The size of the address range associated with BARi is in 4KB increments. Value of 0 signifies BARi is not used.

Note 1: The reset values for each BAR Size as given here:
BAR0: Size = (NV_ADDRESS_MAP_REMAP_PCIE_SIZE >> 12)
BAR1-8: 0 (disabled)

Field	R/W	R/W	Reset	Description
Start	31:12	R/W	See Note 2	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

Note 2: The reset values depend on the system address mapping after reset and are specific for each BAR as follows:
BAR0: (NV_ADDRESS_MAP_REMAP_PCIE_BASE >> 12)
BAR1-8: Start = 0

32.3.1.2 FPCI BARi Mapping

In the register name, i ranges from 0 to 8.

Field	Bits	R/W	Reset	Description
Start	31:4	R/W	See Note 3	The start of the FPCI address space mapped into the BARi range of PCI memory space. Bits 31:4 correspond to FPCI address bits 39:12.

Field	Bits	R/W	Reset	Description
Memory Mapped Access	0	R/W	1	Indicates if the address region is memory mapped to configuration or IO space. 1 **Memory Mapped Access** (PW only) 0 **IO/Config Access** (NWP only)

Note 3: The reset values depend on the system address mapping after reset and are specific for each BAR as follows:
BAR0: Size = (NV_ADDRESS_MAP_REMAP_PCIE_SIZE >> 12)
BAR1-8: Start = 0

32.3.1.3 MSI Bar Size

Field	Bits	R/W	Reset	Description
Size	19:0	R/W	0	The size of the address range associated with MSI BAR is in 4KB increments. Value of 0 signifies BAR is not used.

32.3.1.4 MSI FPCI BAR Start

Field	Bits	R/W	Reset	Description
Start	31:4	R/W	See Note 4	The start of upstream FPCI address space for MSI BAR. The upstream FPCI address is compared to start/4KB range for MSI BAR to determine if the access is MSI. Bits 31:4 of MSI BAR start correspond to upstream FPCI address bits 39:12.

Note 4: The reset values for MSI BAR FPCI Start = (NV_PROJ__PCIE2_RP_MSI_FPCI_BAR)

32.3.1.5 MSI AXI BAR Start

Field	Bits	R/W	Reset	Description
Start	31:12	R/W	See Note 5	The start of upstream AXI address space for MSI BAR. The upstream FPCI address is compared to start/4KB range for MSI BAR to determine if the access is MSI. Bits 31:12 of MSI BAR start correspond to upstream AXI address bits 39:12.

Note 5: The reset values for MSI BAR AXI Start = (NV_ADDRESS_MAP_EMEM_BASE >> 12)

32.3.1.6 MSI VECTORi

In the register name, i ranges from 0 to 7.

Field	Bits	R/W	Reset	Description
VECTORi	31:0	R/W	0	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.

32.3.1.7 MSI ENABLE VECTORi

In the register name, i ranges from 0 to 7.

Field	Bits	R/W	Reset	Description
ENABLE VECTORi	31:0	R/W	0	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.

32.3.1.8 Configuration

Bits	R/W	Reset	Description
16	R	0	PE2 PRESENT. PCIe slot 1 present pin, when 0 indicates a PCIe card is present in the slot
15:14	R/W	10b	Reserved
13	R	0	PE1 PRESENT. PCIe slot 1 present pin, when 0 indicates a PCIe card is present in the slot
12	R	0	PE0 PRESENT. PCIe slot 0 present pin, when 0 indicates a PCIe card is present in the slot
11	R	1	Target Read Idle. This read-only bit provides status reads to AFI target. A value of 1b indicates there are no outstanding reads to downstream FPCI.
10	R	1	Target Write Idle. This read-only bit provides status writes to AFI target. A value of 1b indicates there are no outstanding writes to downstream FPCI.
9	R	1	MSI Vector Empty . This read-only bit provides status on whether MSI vector registers have any active bits.
6:1	R/W	0	Reserved
0	R/W	0	Enable FPCI When the PCI device block is disabled, it is completely invisible on the PCI bus, i.e. it doesn't even process PCI configuration accesses. Note: This bit qualifies device_disable_t0c0, device_disable_t0c1 and device_disable_t0c2 of PCIe 2.0.

32.3.1.9 CLKGATE HYSTERESIS

Bit	R/W	Reset	Description
7:0	R/W	20	CLK_DISABLE_CNT. Number of AFI clock cycles to wait after clock gating criteria is met to disable AFI/FPCI clocks

32.3.1.10 PLLE

Bit	R/W	Reset	Description
9	R/W	0	**Bypass PADS2PLLE Control.** Overrides PCIe PADS CLKREQ control of PLLE.
8	R/W	0	**Bypass PCIE2PLLE Control.** Overrides PCIe 2.0 CLOCK CLAMP control of PLLE.
7:2	R	0	Reserved

Bit	R/W	Reset	Description
1	R/W	0	**Enable PADS2PLLE Control**. When active all CLKREQ going inactive signal disable to PLLE. When SATA, PCIe 2.0 and PADS all signal DISABLE to PLLE then PLLE will disable.
0	R/W	0	**Enable PCIE2PLLE Control**. When active PCIe 2.0 CLOCK CLAMP going active signal disable to PLLE. When SATA, PCIe 2.0 and PADS all signal DISABLE to PLLE then PLLE will disable.

32.3.1.11 FPCI Error Masks

Bit	R/W	Reset	Description
2	R/W	0	**Master Abort Mask.** This bit allows FPCI error to be forwarded to AXI response when FPCI error response indicates Master Abort. 1 = forward error, 0 = return AXI OKAY response (2'b0)
1	R/W	0	**Data Error Mask.** This bit allows FPCI error to be forwarded to AXI response when FPCI error response indicates Data Error. 1 = forward error, 0 = return AXI OKAY response (2'b0)
0	R/W	0	**Target Abort Mask**. This bit allows FPCI error to be forwarded to AXI response when FPCI error response indicates Target Abort or if FPCI does not respond with devsel to the target access. 1 = forward error, 0 = return AXI OKAY response (2'b0)

32.3.1.12 Interrupt Masks

Bit	R/W	Reset	Description
8	R/W	0	**MSI Mask.** MSI to MPCORE gated by mask
7:1	R	0	Reserved
0	R/W	0	**AFI Interrupt Mask.** AFI Interrupt to MPCORE gated by mask.

32.3.1.13 Interrupt Control

Detected errors are logged in an error signature register as specified here. There is one error signature register that always store the first error that occurred after enabling the error capture. The interrupt code register logs the interrupt ID shown in the ID column of the table below. This register will store the ID for the associated interrupt when current ID in the register is zero. Software must write zero to the register to re-enable it to store subsequent interrupts. For ID=6, the capture information is the sideband message bits. There is an associated interrupt generated to the CPU. Since FPCI address is 40 bits, a separate capture register is used for the upper 8 bits of the address.

Bit	R/W	Reset	Description
3:0	R/W	0	**Interrupt Code.** Six interrupts codes at this time, see the error and message table below. If the code is 0, logging of the next interrupt is enabled.

Error/Message	ID	Capture Information
AXI Slave error	1	Bits[31:2] of Initiator AXI address, direction of transaction
AXI Decode error	2	
AXI Slave error – PCIe target abort or data error	3	Bits[39:2] of Target FPCI address, direction of transaction
AXI Decode error – PCIe master abort	4	
AXI Decode error – write to NPW address region > 4B	5	Bits[39:2] of Target AXI address, direction of transaction

Error/Message	ID	Capture Information
PCIe 2.0 Sideband message	6	Sideband UnitID, Bits[4:0] of sideband message
FPCI Decode error – not in PCIe 2.0 memory space	7	Bits[39:2] of Target FPCI address, direction of transaction
AXI Decode error – not in AFI BAR or register space	8	Bits[31:2] of Target AXI address, direction of transaction
FPCI Timeout	9	None
Slot Present Pin Change	10	Bits[2:0] capture status of slots 2-0 present pin
Slot Clock Request Change	11	Bits[2:0] capture status of slots 2-0 clock request
TMS Clock Clamp Change	12	Bit[0] captures level of clock clamp
TMS Ready for power down	13	None – all 3 controllers are ready for power down
Peer-to-peer error	14	Error response from endpoint device (tms02ufa_cmd_error), Bits[39:2] of Target FPCI address, direction of transaction

32.3.1.14 Interrupt Signature

Bit	R/W	Reset	Description
31:2	R/W	0	**Interrupt Information.** If signature type is 1 – 5, information bits indicate address bits [31:2], either in FPCI memory space or AXI space. If interrupt code is 6, interrupt signature bits[x:0] contain sideband information {sideband unitid, 3'b0, tms02sm_msg[4:0]}.
1	R	0	Reserved
0	R/W	0	**Direction.** Indicate direction of the AXI transaction. 1 = read 0 = write

32.3.1.15 Interrupt Upper FPCI Address

Bit	R/W	Reset	Description
17:16	R/W	0	**Interrupt Address.** These 2 bits are for the captured endpoint device error response (tms02ufa_cmd_error) when interrupt code is 14.
15:8	R	0	Reserved
7:0	R/W	0	**P2P Error Response.** These 8 bits are the upper byte of captured FPCI address (bits[39:32]) when interrupt code is 3, 4 or 7.

32.3.1.16 Sideband Message Interrupt Enable

The PCI Express block has a sideband interface to transmit System Management and Interrupt messages. The message interface is simple, a message request along with message type and message data are presented to the AXI to FPCI wrapper. The AFI will capture the message in the signature register and assert an interrupt to MPCORE if the associated enable bit of the message enable register is active.

The message is encoded in bits of the request message signal (tms02sm_msg[4:0]). A table of the meaning of the bits is listed below.

The Tegra 3 devices merge PCI Express merge multiple PCI Express Root Ports into one module. To allow the external logic to distinguish which Root Port is the source of the event, the UnitID of the Root Port will be also sent with the message

(tms02sm_unitid[4:0]). Having knowledge of which Root Port is the source of the event is required for system level error handling and interrupt isolation in virtualized systems.

Bit	R/W	Reset	Description
14:0	R/W	0	**Enable Message.** Each of the bits in this register correspond to enabling the associated message shown below

[1:0]	[3:2]	[4]	Enable bit	MSG_TYPE	Description
00	00	1	0	Interrupt	INTA Assertion.
	01		1		INTB Assertion.
	10		2		INTC Assertion.
	11		3		INTD Assertion.
	00	0	4		INTA Deassertion
	01		5		INTB Deassertion
	10		6		INTC Deassertion
	11		7		INTD Deassertion
01	00	0	8	Error	Correctable Error
	01		9		Un-Correctable Non-Fatal Error
	10		10		Un-Correctable Fatal Error
10	00	0	11	PME	PME assertion.
10	01	0	12	Hotplug	Hotplug SCI assertion
11	00	1	13	Interrupt	Root Port Interrupt Assertion
11	00	0	14		Root Port Interrupt Deassertion

32.3.1.17 AFI Interrupt Enable

Bit	R/W	Reset	Description
12	R/W	0	**EN_P2P_ERR.** Enable bit for interrupt code 14 in the Errors and Messages table.
11	R/W	0	**EN_RDY4PD_SENSE.** Enable bit for interrupt code 13 in the Errors and Messages table.
10	R/W	0	**EN_CLKCLAMP_SENSE.** Enable bit for interrupt code 12 in the Errors and Messages table.
9	R/W	0	**EN_PE_CLKREQ_SENSE.** Enable bit for interrupt code 11 in the Errors and Messages table.
8	R/W	0	**EN_PE_PRSNT_SENSE.** Enable bit for interrupt code 10 in the Errors and Messages table.
7	R/W	0	**EN_FPCI_TIMEOUT.** Enable bit for interrupt code 9 in the Errors and Messages table.
6	R/W	0	**EN_AXI_DECERR.** Enable bit for interrupt code 8 in the Errors and Messages table.
5	R/W	0	**EN_DFPCI_DECERR.** Enable bit for interrupt code 7 in the Errors and Messages table.
4	R/W	0	**EN_TGT_WRERR.** Enable bit for interrupt code 5 in the Errors and Messages table.
3	R/W	0	**EN_TGT_DECERR.** Enable bit for interrupt code 4 in the Errors and Messages table.
2	R/W	0	**EN_TGT_SLVERR.** Enable bit for interrupt code 3 in the Errors and Messages table.
1	R/W	0	**EN_INI_DECERR.** Enable bit for interrupt code 2 in the Errors and Messages table.
0	R/W	0	**EN_INI_SLVERR.** Enable bit for interrupt code 1 in the Errors and Messages table.

32.4 PCIe Root Port Control Registers

The registers mentioned here are registers are used to drive static configuration signals to PCIe root port controller.

32.4.1.1 FPCI Timeout

Bit	R/W	Reset	Description
31	R/W	0	**Timeout Threshold.** SM (system management) threshold specifying how long to wait for response from FPCI before declaring it timeout
30:20	R	0	Reserved
19:0	R/W	0	**Timeout Enable.** SM (system management) timeout enable

32.4.1.2 PCIe Throttle

Bit	R/W	Reset	Description
31	R/W	0	**Throttle Enable.** SM (system management) to PCIe throttle enable. When enabled, the throttle period and duty cycle are used in place of those in the vendor defined thermal management config register.
30:16	R	0	Reserved
15:4	R/W	0	**Throttle Period.** SM (system management) to PCIe override to root port vendor defined thermal management period register.
3	R	0	Reserved
2:0	R/W	0	**Throttle Duty Cycle.** SM (system management) to PCIe override to root port vendor defined thermal management duty cycle register.

32.4.1.3 PME

Bit	R/W	Reset	Description
19	R	0	**C2 Presence State.** PCIe Controller 2 Link Presence State
18	R	0	**C2 PME Ack.** PCIe Controller 2 Endpoint PME Ack
17	R	0	**C2 PME.** PCIe Controller 2 Endpoint PME message
16	R/W	0	**C2 PME Turn Off.** SM (system management) to PCIe Controller 2 PME Turn Off. This field should remain low for Tegra 3 devices.
15:12	R	0	Reserved
11	R	0	**C1 Presence State.** PCIe Controller 1 Link Presence State
10	R	0	**C1 PME Ack.** PCIe Controller 1 Endpoint PME Ack
9	R	0	**C1 PME.** PCIe Controller 0 Link Presence State
7	R	0	Reserved
8	R/W	0	**C1 PME Turn Off.** SM (system management) to PCIe Controller 1 PME Turn Off. This field should remain low for Tegra 3 devices.
6	R	0	**C0 Presence State.** PCIe Controller 0 Link Presence State
5	R	0	**C0 PME Ack.** PCIe Controller 0 Endpoint PME Ack

Bit	R/W	Reset	Description
4	R	0	**C0 PME.** PCIe Controller 0 Endpoint PME message
3:1	R	0	Reserved
0	R/W	0	**C0 PME Turn Off.** SM (system management) to PCIe Controller 0 PME Turn Off. This field should remain low for Tegra 3 devices.

32.4.1.4 Request Pending

Bit	R/W	Reset	Description
11	R	0	**C2 Non-ISO Request Pending.** SM (system management) status that Non-ISO request pending from PCIe Controller 2 to FPCI when in low power mode.
10	R	0	**C2 ISO Request Pending.** SM (system management) status that ISO request pending from PCIe Controller 2 to FPCI when in low power mode.
9	R	0	**C2 Non-Coherent Request Pending.** SM (system management) status that Non-coherent request pending from PCIe Controller 2 to FPCI when in low power mode.
8	R	0	**C2 Coherent Request Pending.** SM (system management) status that coherent request pending from PCIe Controller 2 to FPCI when in low power mode.
7	R	0	**C1 Non-ISO Request Pending.** SM (system management) status that Non-ISO request pending from PCIe Controller 1 to FPCI when in low power mode.
6	R	0	**C1 ISO Request Pending.** SM (system management) status that ISO request pending from PCIe Controller 1 to FPCI when in low power mode.
5	R	0	**C1 Non-Coherent Request Pending.** SM (system management) status that Non-coherent request pending from PCIe Controller 1 to FPCI when in low power mode.
4	R	0	**C1 Coherent Request Pending.** SM (system management) status that coherent request pending from PCIe Controller 1 to FPCI when in low power mode.
3	R	0	**C0 Non-ISO Request Pending.** SM (system management) status that Non-ISO request pending from PCIe Controller 0 to FPCI when in low power mode.
2	R	0	**C0 ISO Request Pending.** SM (system management) status that ISO request pending from PCIe Controller 0 to FPCI when in low power mode.
1	R	0	**C0 Non-Coherent Request Pending.** SM (system management) status that Non-coherent request pending from PCIe Controller 0 to FPCI when in low power mode.
0	R	0	**C0 Coherent Request Pending.** SM (system management) status that coherent request pending from PCIe Controller 0 to FPCI when in low power mode.

32.4.1.5 PCIe CONFIG

Bit	R/W	Reset	Description
28:24	R/W	1	**T0C2 Unit ID.** PCIe Controller 2 Unit ID. Upstream FPCI request unit ID from controller 2.
23:20	R/W	4	**XBAR Config**. SM (system management) configuration of PCIe crossbar. This input determines how PCIe root port connects to PCIe endpoints. Valid configurations for Tegra 3 are : • 1x4,1x2 • 1x4,2x1 • 3x2
16:12	R/W	3	**T0C1 Unit ID.** PCIe Controller 1 Unit ID. Upstream FPCI request unit ID from controller 1.

Bit	R/W	Reset	Description
8:4	R/W	2	**T0C0 Unit ID.** PCIe Controller 0 Unit ID. Upstream FPCI request unit ID from controller 0.
3	R/W	1	**Disable Device T0C2.** Disable PCIe controller C2. This bit is qualified with Enable FPCI to generate PCIe 2.0 signal ***device_disable_t0c2***.
2	R/W	1	**Disable Device T0C1.** Disable PCIe controller C1. This bit is qualified with Enable FPCI to generate PCIe 2.0 signal ***device_disable_t0c1***.
1	R/W	0	**Disable Device T0C0.** Disable PCIe controller C0. This bit is qualified with Enable FPCI to generate PCIe 2.0 signal ***device_disable_t0c0***.
0	R	1	Reserved

32.4.1.6 REV ID

Bit	R/W	Reset	Description
28:24	R/W	0	**REVID Write Enable.** Write Enable for PCI backdoor rev ID override value.
23:20	R/W	0	**REVID Override.** Override for PCI config revision id read-only register. This allows backdoor changes to rev ID for metal spins.

32.4.1.7 Top-of-Memory (TOM)

Bit	R/W	Reset	Description
29:16	R/W	3F3Fh	Top of Memory Limit 2. Determines peer-to-peer range as: {TOM2:: 26'b0} to 0xFC_FFFF_FFFF
5:0	R/W	3Fh	Top of Memory Limit 1. Determines peer-to-peer range as: {TOM1 :: 26'b0} to 0xFFFF_FFFF (except MSI region)

32.4.1.8 P2PBOM CTRL

Bit	R/W	Reset	Description
19:0	R/W	0	**P2P_BASE.** Peer-to-peer Bottom of Memory drives bits 39:20 of p2p_base_63_to_20 with top bits tied to 0.

32.4.1.9 P2PTOM CTRL

Bit	R/W	Reset	Description
19:0	R/W	003FFh	**P2P_LIMIT.** Peer-to-peer Bottom of Memory drives bits 39:20 of p2p_limit_63_to_20 with top bits tied to 0.

32.4.1.10 FUSE

Bit	R/W	Reset	Description
14:12	R/W	3	**PCIe Width T0C2.** Configure PCIe controller 1 as x1, x2, x4, x8, or x16. This should remain 3'b011 for Tegra 3 single endpoint or 3'b010 for multiple endpoints.
11	R		Reserved
10:8		3	**PCIe Width T0C1.** Configure PCIe controller 1 as x1, x2, x4, x8, or x16. This should remain 3'b011 for Tegra 3 single endpoint or 3'b010 for multiple endpoints.

Bit	R/W	Reset	Description
7	R		Reserved
6:4		3	**PCIe Width T0C0.** Configure PCIe controller 0 as x1, x2, x4, x8, or x16. This should remain 3'b011 for Tegra 3 single endpoint or 3'b010 for multiple endpoints. The widths are enumerated below. • X1_MODE – 3'b001 • X2_MODE – 3'b010 • X4_MODE – 3'b011 • X8_MODE – 3'b100 • X16_MODE – 3'b101
3	R		Reserved
2		1	**PCIe Gen2 Disable.** Disable Gen 2 capability of PCIe. This should remain on for Tegra 3.
1		1	**PCIe SLI Disable.** Disable SLI capability for GPU. This bit controls peer-to-peer transactions. For single controller configuration, this should remain on. For dual controller configuration, this should be disabled when enabling peer-to-peer transactions.
0	R		Reserved

32.4.1.11 C0-C1 PMU

Bit	R/W	Reset	Description
24	R	0	**C1 PMU Toggle.** PCIe Controller 1 PMU toggle response from PCIe
23:20	R	0	**C1 PMU Status.** PCIe Controller 1 PMU Status
19:17	R		Reserved
16	R	0	**C0 PMU Toggle.** PCIe Controller 0 PMU toggle response from PCIe
15:12	R	0	**C0 PMU Status.** PCIe Controller 0 PMU Status
11:8	R/W	0	**C1 Load Indicator Scale.** PCIe Controller 1 PMU Load Indicator Scale
7:4	R/W	0	**C0 Load Indicator Scale.** PCIe Controller 0 PMU Load Indicator Scale
3:1	R		Reserved
0	R/W	0	**Update Fast Toggle.** PMU Load Indicator Enable.

32.4.1.12 C2 PMU

Bit	R/W	Reset	Description
16	R	0	**C2 PMU Toggle.** PCIe Controller 2 PMU toggle response from PCIe
15:13	R		Reserved
11:8	R	0	**C2 PMU Status.** PCIe Controller 2 PMU Status
7:4	R/W	0	**C2 Load Indicator Scale.** PCIe Controller 2 PMU Load Indicator Scale
3:0	R		Reserved

32.4.1.13 PCIe Clock Config/Status

Bit	R/W	Reset	Description
27:24	R	0	**C2 Pad Macro CLK Select.** Clock select to pad macro from PCIe Controller 2. For Tegra 3 devices, this should remain 0.
23:20	R	0	**C1 Pad Macro CLK Select.** Clock select to pad macro from PCIe Controller 1. For Tegra 3 devices, this should remain 0.
19:16	R	0	**C0 Pad Macro CLK Select.** Clock select to pad macro from PCIe Controller 0. For Tegra 3 devices, this should remain 0.
15	R	0	**C2 Select XTXCLK1X Gen2.** Request to select Gen2 speed clock (500 MHz) for PCIe Controller1 XTXCLK1X. This is generated when register settings for PCIe 2.0 specify Gen2 speed clocks. This should remain low for Tegra 3 devices.
14	R	0	**C2 Disable XTXCLK1X.** Request to gate PCIe Controller 2 XTXCLK1X when in low power mode.
13	R	0	**C1 Disable XTXCLK1X.** Request to gate PCIe Controller 1 XTXCLK1X when in low power mode.
12	R	0	**C0 Disable XTXCLK1X.** Request to gate PCIe Controller 0 XTXCLK1X when in low power mode.
11	R	0	**Clamp CLK.** Request to gate TMS/FPCI clocks when in low power mode.
10	R	0	**C1 Select XTXCLK1X Gen2.** Request to select Gen2 speed clock (500 MHz) for PCIe Controller1 XTXCLK1X. This is generated when register settings for PCIe 2.0 specify Gen2 speed clocks. This should remain low for Tegra 3 devices.
9	R	0	**C0 Select XTXCLK1X Gen2.** Request to select Gen2 speed clock (500 MHz) for PCIe Controller0 XTXCLK1X. This is generated when register settings for PCIe 2.0 specify Gen2 speed clocks. This should remain low for Tegra 3 devices.
8	R	0	**Select XCLK Gen2.** Request to select Gen2 speed clock (500 MHz) for XCLK. This is generated when register settings for PCIe 2.0 specify Gen2 speed clocks. This should remain low for Tegra 3.
6	R/W	0	**C2 Off XTXCLK1X.** Controller 2 Acknowledge to Disable XTXCLK1X request. Used for clock gating.
5	R/W	0	**C2 Ready XTXCLK1X Gen2.** Controller 2 Acknowledge to Select XTXCLK1X Gen2 request. This should remain low for Tegra 3.
4	R/W	0	**C1 Off XTXCLK1X.** Controller 1 Acknowledge to Disable XTXCLK1X request. Used for clock gating.
3	R/W	0	**C1 Ready XTXCLK1X Gen2.** Controller 1 Acknowledge to Select XTXCLK1X Gen2 request. This should remain low for Tegra 3.
2	R/W	0	**C0 Off XTXCLK1X.** Controller 0 Acknowledge to Disable XTXCLK1X request. Used for clock gating.
1	R/W	0	**C0 Ready XTXCLK1X Gen2.** Controller 0 Acknowledge to Select XTXCLK1X Gen2 request. This should remain low for Tegra 3.
0	R/W	0	**Ready XCLK Gen2.** Acknowledge to Select XCLK Gen2 request. This should remain low for Tegra 3.

32.4.1.14 Bus Trace

Bit	R/W	Reset	Description
21	R	0	**Bus Trace Trigger.** Bus trigger from bus trace module in PCIe 2.0
20	R/W	0	**Trace External Start.** Start signal to bus trace module in PCIe 2.0
19:18	R		Reserved
17:16	R/W	0	**Bus Trace Mux Sel.** Mux select to bus trace module in PCIe 2.0
15:0	R/W	0	**Chip ID.** Chip ID to bus trace module in PCIe 2.0

32.5 PCIe Slot Registers

The registers mentioned here are registers are used to control clock and reset to each of the PCIe slots.

32.5.1.1 PEX0 CTRL

Bit	R/W	Reset	Description
3	R/W	0	**PEX0_REFCLK_EN.** Clock Enable to PCIe slot 0
2	R		Reserved
1	R/W	0	**PEX0_CLKREQ_EN.** Enable to allow CLKREQ0 to control PCIe slot 0 clock
0	R/W	0	**PEX0_RST_L**. Reset to PCIe slot 0

32.5.1.2 PEX0 STATUS

Bit	R/W	Reset	Description
0	R	0	**PEX0_CLKREQ_L**. Clock request status for PCIe slot 0

32.5.1.3 PEX1 CTRL

Bit	R/W	Reset	Description
3	R/W	0	**PEX1_REFCLK_EN.** Clock Enable to PCIe slot 1
2	R		Reserved
1	R/W	0	**PEX1_CLKREQ_EN.** Enable to allow CLKREQ1 to control PCIe slot 1 clock
0	R/W	0	**PEX1_RST_L**. Reset to PCIe slot 1

32.5.1.4 PEX1 STATUS

Bit	R/W	Reset	Description
0	R	0	**PEX1_CLKREQ_L**. Clock request status for PCIe slot 1

32.5.1.5 PEX2 CTRL

Bit	R/W	Reset	Description
3	R/W	0	**PEX2_REFCLK_EN.** Clock Enable to PCIe slot 2
2	R		Reserved
1	R/W	0	**PEX2_CLKREQ_EN.** Enable to allow CLKREQ1 to control PCIe slot 2 clock
0	R/W	0	**PEX2_RST_L**. Reset to PCIe slot 2

32.5.1.6 PEX2 STATUS

Bit	R/W	Reset	Description
0	R	0	**PEX2_CLKREQ_L**. Clock request status for PCIe slot 2

32.5.1.7 PEXBIAS CTRL

Bit	R/W	Reset	Description
0	R/W	1	**PEX_BIAS_PWRD**. Power down to PCIe slot clock bias pad

[THIS PAGE INTENTIONALLY LEFT BLANK]

33.0 I2C CONTROLLER

The I^2C controller (I2C) implements an I^2C 2.1 specification-compliant I2C master and a slave controller. I2C supports multiple masters and slaves. It also supports fast mode (400 KHz operation) and high speed mode (3.4 MHz operation). Tegra® 3 devices have five instances of this controller. All five instances present identical I2C master functionality.

I2C supports DMA over APB bus in addition to single byte transfers. I2C also supports packet mode transfers where the data to be transferred is encapsulated in a predefined packet format as payload and sent to I2C 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 and fast mode of operation (0-400 KHz) as well as high speed mode (up to 2.577 MHz).
- 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 kilobytes of transfer in packet mode. Can be extended beyond 4Kbytes 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 general call addressing
- Supports Recognition and Transfer of data to peripherals that do not send an acknowledge
- Slave can also be configured not to acknowledge
- Master supports packet based DMA

Clocking

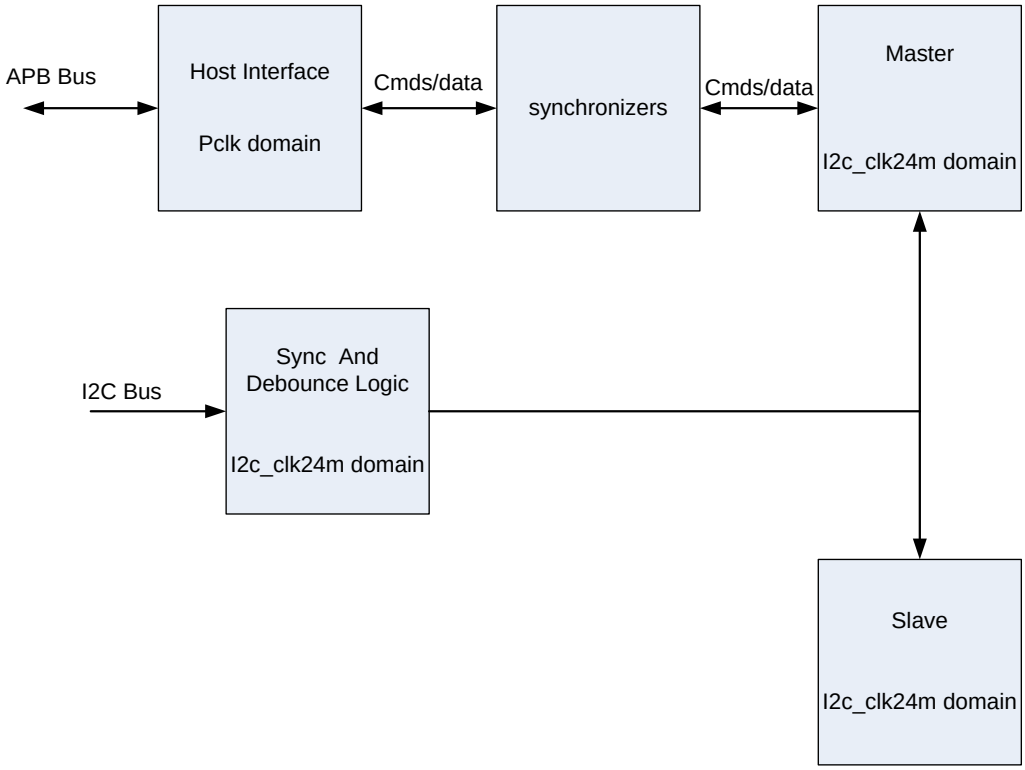
I2C has two clock domains:

- Host interface runs on APB clock. APB clock is derived from sys_clk and can be 1.0, ½, 1/3 or ¼ times the sys_clk.
- I2C interface controller runs on a fixed frequency clock running at 72 MHz. The 72 MHz clock is derived from PLLP. Even though we can mux between PLLP, PLLC, PLLM, and OSC clocks, PLLP is always selected and used.

33.1 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 83 below shows the I2C controller in both the master and slave modes.

Figure 83 I2C Controller Block Diagram



33.2 Interfaces

A typical peripheral controller might require the following set of steps to communicate or control an external peripheral:

1. Set up the registers
2. Program the ‘go’ bit to start the interaction with external peripheral
3. Peripheral controller collects the response from peripheral and interrupts software
4. Software reads the response

33.2.1 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 hardware peripheral controller and software as described in section 33.2 can be classified into:

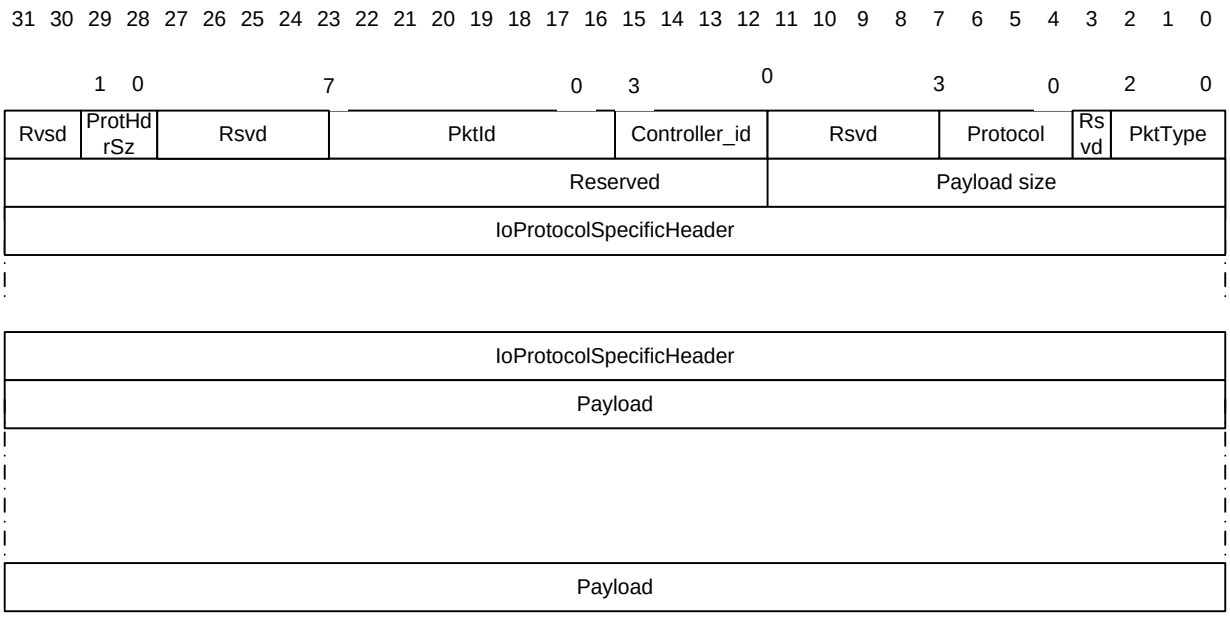
- ‘Request’ flow which can represent the register writes
- ‘Response’ flow for peripheral controller responses sent back to software, and
- ‘Interrupt’ flow for out-of-band handshake between hardware and software

Packets within a flow can be exchanged in a manner that requires full intervention from CPU (PIO mode) or least intervention from CPU (DMA mode). The I/O packet described below is a structure that can be used to represent various flows in the system.

33.2.2 I/O Packet Format

An I/O Packet consists of header and payload. The header consists of two parts: Two words of generic header, and one to four words of protocol specific header.

Figure 84 I/O Packet



33.2.2.1 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 85. 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. 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
3:3	Reserved	Reserved

Bits	Field Value	Description
2:0	PktType	Packet type information 0 = request 1 = response 2 = interrupt 3 = stop 4-7 = reserved

33.2.2.2 I/O Packet Word 1

Word 1 of the I/O packet contains the Generic Header Word 1. The contents of generic header word 1 are described in the table below.

Table 86. 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 send transmit data, the total packet size is payload_size + 8 bytes of generic header + protocol specific header.

However for request packets with read command, 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.

33.2.3 Transferring Packets

Packets can be transferred using PIO or DMA modes. A peripheral controller can choose to implement one or both modes. 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.

33.2.4 I/O 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) a particular implementation chooses.

I2C master supports two kinds of packets.

- Transmit packets that flow from Host to Controller.
- Receive data packets that flow from Controller to Host.

I2C has one word of Protocol Specific Header. The contents of I2C protocol specific header are described in the table below.

Table 87. I/O Protocol Specific Header

Bit	Name	Description
31:26	Reserved	Reserved
25	RESP_PKT_FREQ	Response packet frequency. This field is valid only if I2C master is transmitting (READ/WRITE field below). If it is receiving then response packets will be 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.

Bit	Name	Description
		0: Don't 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 (3.4MHz 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 indicates to put a repeat start condition on the bus(to continue transaction) 0 indicates to put a stop condition on the bus
15	ContinueXfer	This bit overrides 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

33.3 Programming Guidelines

33.3.1 I2C Frequency Divisor Register

The FREQUENCY DIVISOR register (CLK_SOURCE_I2C register, whose address is: 0x600060A4) must be programmed as a function of the CLK_SOURCE Selected for I2C as follows:

```
I2C_CLK = CLK_SOURCE.I2C / (8 * I2C FREQUENCY DIVISOR)
```

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 as rise and fall delays of 300ns per the I^2C specification.

The clock enable (bit-12 of CLK_OUT_ENB.L register) must also be given to I2C controller, before any of the registers are written.

33.3.2 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. The 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].

33.3.3 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. I2C Master Controller can be programmed to transact with both 7-bit and 10-bit addressed slaves. The bit [0] of 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 the bits 6 & 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].

33.3.4 I2C Data Registers

There are two data registers, I2C_CMD_DATA1and I2C_CMD_DATA2, each of 32 bit length, which are to be loaded with the data to be transmitted or received as an I^2C Master. When used as 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).

33.3.5 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 bit I2C_CNGF [9] is used to issue a “Begin-Transaction” command to the I2C 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 the bits [8:0] of I2C_CNFG register before the bit I2C_CNFG [9] is programmed to one.

33.3.6 I2C Controller STATUS Register

This register (I2C_STATUS) gives the status of the I^2C Master operation. It includes the busy status of the I^2C Bus and the completion status of the command executed.

33.3.7 I2C Controller Slave Status Register

This register (I2C_SL_STATUS) contains the status bits as I^2C Slave. ,

Tegra 3 contains status bits as follows:

- Receive Interrupt
- New Transaction Received
- Direction of transfer.
- Bit 4 contains END_TRANS bit. This bit is set when an I^2C transfer is complete (as indicated by either a stop or repeat start condition).
- Bit 5 contains RST_SL. The byte received following the receipt of general call address is 0x06. Therefore programmable part of the slave address needs to be reset and reprogrammed.
- Bit 6 contains REPROG_SL. The byte received following the receipt of general call address is 0x04. Therefore reprogram the programmable part of slave address.
- Bit 7 contains HW_MSTR_INT. When set this bit indicates that master address has been transmitted after 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.

Note that the I^2C Controller generates an interrupt at the completion of each data byte received. In response to this interrupt, firmware must read 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^2C Controller will delay the release of the ACK handshake on the I^2C bus until the SL.IRQ bit has been cleared by this firmware read operation.

I2C_SL_STATUS needs to be either 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 Tegra 3 as a slave is **receiving** data,

- I2C_SL_STATUS.SL_IRQ bit being set indicates a data byte has been received.
- Read to I2C_SL_STATUS clears SL_IRQ and returns the transfer direction in RNW field.
- Data byte is already present in I2C_SL_RCVD register. This can be read which causes the bus to be released.
- SL_IRQ is set again upon reception of next byte

If Tegra 3 as a slave is **transmitting** data,

- I2C_SL_STATUS.SL_IRQ bit being set indicates that address has been received.
- Read to I2C_SL_STATUS clears SL_IRQ and returns the transfer direction in RNW field.
- Data byte needs to be written to 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

33.3.8 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 of 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 (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 85 Interrupt Generation: Master Transmits to Internal Slave

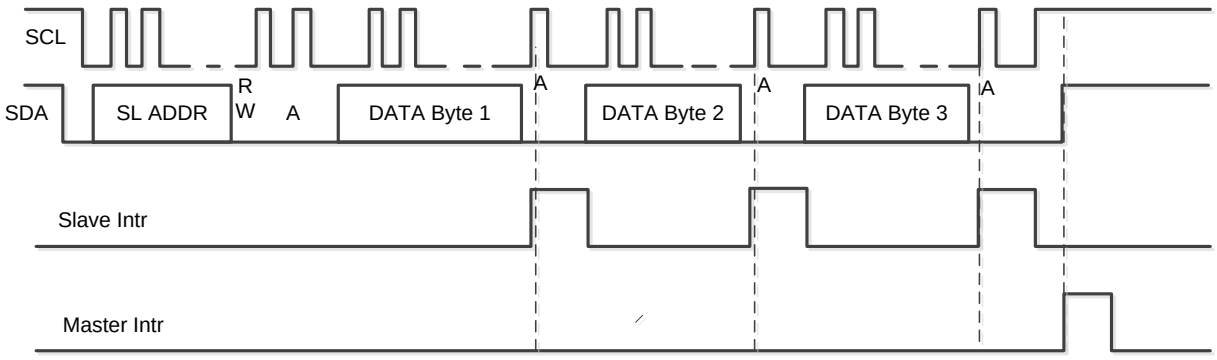
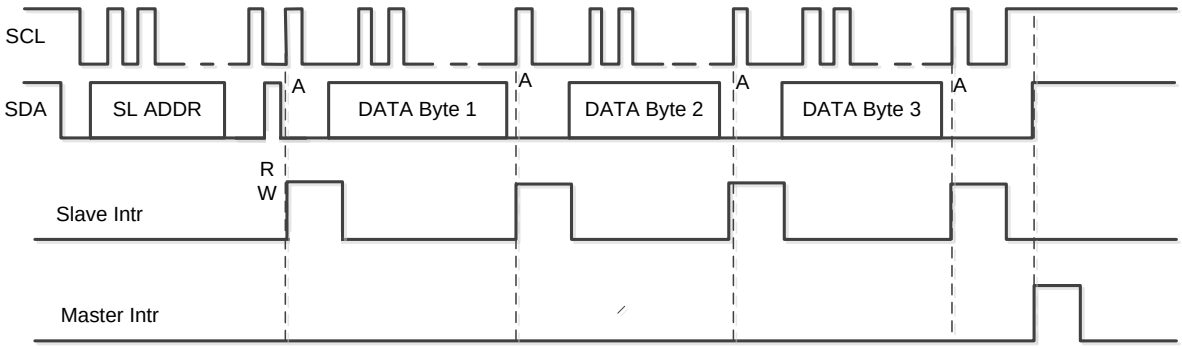


Figure 86 Interrupt Generation: Master Reads from Internal Slave



33.4 Programming Guidelines for Packet Based Interface

33.4.1 Packet Based Interface Registers

The following registers need to be programmed in case of packet based interface:

- I2C TX Packet FIFO Register
- 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
- INTERRUPT_STATUS_REGISTER
- PACKET_MODE_EN , DEBOUNCE_CNT and NEW_MASTER_FSM field of I2C Controller Configuration Register

All other registers/fields have no meaning in packet mode and should be used only normal mode.

33.4.2 Programming Packet Header and Payload

SW should program the I2C_TX_PACKET_FIFO register. It writes packet header followed by payload. Header size is variable, could vary from 3 to 5 words. For I2C, it is 3 words for request packets and 4 words for response packets. The first two words of the header contain generic header. The third word of the Packet (and 4^{th} as well in case of response packet) contains I2C transaction specific information. Payload contains the actual data to be written to the slave. In case of read operation, payload is nil, and hence, packet contains only header.

33.4.3 Reading from Rx_fifo

The data received on the bus is pushed into rx_fifo. In PIO mode A, a request interrupt is generated when the attention level is reached and SW reads rx_fifo register to get the data. In DMA mode, a DMA trigger is asserted after reaching the attention level.

33.4.4 Error Handling

In case of an error due to nack or loss of arbitration or txfifo overflow or rxfifo underflow, the following steps describe what happens when an error occurs:

- A stop condition is put on the bus and interrupt is generated
- Status register is updated with the current error packet id, and the byte number at which the error occurred.
- SW should reset the controller
- In case of 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 and DMA need to be restarted (if using DMA transfer) 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.

Repeat start bit in packet mode can complicate the situation further since when back to back packets are sent to a slave with repeat start bit set, and an error occurs then we can't reliably know during which packet transfer the error occurred. Therefore the entire stream of packets that were sent back to back using repeat start need to be sent again.

On the receive case it is a bit simpler since if an error occurs, and bytes for a packet have already been received then that packet can be deemed complete. Only incomplete transfers need to be retried.

33.4.5 Programming repeat_start/stop

This field indicates whether 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. Repeat_start/stop bit can be used for combining read operations and write operations within a single transaction with repeat start, or to do transfers beyond the 4Kbyte limit. This bit is present in the protocol specific header.

Tegra 3 supports transfer size beyond 4Kbytes without repeat start condition, again by combining multiple packets. This is in addition to ability to use repeat start condition to do continuous transfers over 4Kbyte limit.

33.4.6 FIFO Control

In DMA mode, TX_FIFO_TRIG indicates the number of words that need to be empty in TX FIFO for the DMA trigger to be asserted. RX_FIFO_TRIG indicates the number of words that need to be full in RX FIFO for the DMA trigger to be asserted.

In PIO mode, TFIFO_DATA_REQ in 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 INTERRUPT_STATUS_REGISTER will be set if RX FIFO full count is more than the value programmed in RX_FIFO_TRIG.CPU will be interrupted if status bits are set and their corresponding INT_EN is set in the INTERRUPT_MASK_REGISTER. The depth of TX and RX FIFOs is 8.

33.4.7 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).

33.4.8 Example Transfer

A programming example for 1 byte read followed by 5 byte write

1. Program the PACKET_MODE_EN field to 1
2. Write the generic packet header
3. Write payload size = 0x0(which means 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(which means 5 bytes)
7. Write the i2c specific header read/write = 0 and repeat_start/stop
8. Write the 12 bytes(word aligned)

33.4.9 Removal of Register read side effects

Tegra 3 implements packet mode transfers for I2C slave. In packet mode, slave holds off ACK on the very last byte of transfer until the host explicitly indicates to I2C to release the bus by ACKing the last byte. The Host is responsible for providing this indication to I2C slave in a timely fashion before the SMBUS timeouts happens.

After checking the packet contents Host needs to send a control packet with:

- "ACK_LAST_BYTE" in protocol specific header set to 1'b1

- PAYLOAD_SIZE in generic packet header set to zero
- “READ/WRITE” bit in protocol specific header set to 1’b1 (READ).

Slave will release the bus as soon as it receives this control packet.

33.4.10 DMA capability for I2C slave

Packet mode capability of the slave combined with DMA transfer capabilities eliminates the need for byte by byte interaction between driver and hardware.

33.4.11 Programming Guidelines for Slave

The data is transmitted/received via single register **I2C_I2C_SL_RCVD_0**

To work in Tegra 3 mode where we can transfer one byte at a time

- Set I2C_I2C_SL_CNFG_0. ENABLE_SL

In Tegra 3, data can be transmitted/received through **I2C_I2C_SLV_TX_PACKET_FIFO_0** and **I2C_I2C_SLV_RX_FIFO_0** registers respectively. For this, FIFO_XFER_EN bit in **I2C_I2C_SL_CNFG_0** register needs to be set.

In fifo xfer mode, slave by default works in one byte mode, i.e initially when a single byte is received, ack is withheld and interrupt is generated. SW should indicate whether to ack this byte or not by writing into ACK_LAST_BYTE field of **I2C_I2C_SL_CNFG_0** register. Please note that ACK_LAST_BYTE_VALID field also needs to be set to indicate HW that ACK_LAST_BYTE is being updated. When SW receives interrupt due to SLV_RX_BUFFER_FILLED, it can check ACK_WITHHELD of I2C_**I2C_SLV_PACKET_STATUS_0** register to see if slave is withholding the ack for the last byte.

SW can update the buffer size from default to any number by programming BUFFER_SIZE or by writing a packet where it updates payload size.

Packet structure consists of header followed by data in case of Rd operation, whereas it consists of only header in case of Wr operation. Packets should be programmed into **I2C_I2C_SLV_TX_PACKET_FIFO_0** register .The data received will be stored in **I2C_I2C_SLV_RX_FIFO_0** register. When SW reads from this register, it causes pop.

Packet header consists of 3 Words, generic header, payload size, and protocol specific header.

Packet Wr operation works as follows.

After enabling packet

To work in Tegra 3 packet mode for Wr 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 interrupt SLV_RX_BUFFER_FILLED.By default buffer_size is one and interrupt will be generated after first byte and ack is withhold.
- Clear the interrupt and indicate whether to ack the last byte received by writing into I2C_I2C_SL_CNFG_0. ACK_LAST_BYTE_VALID and into I2C_I2C_SL_CNFG_0. ACK _LAST_BYTE
- Write Generic Packet Header
- Write Payload size
- Write protocol Specific header.

  The buffer_size will updated with payload size set in packet header. Interrupt will be generated again after receiving no.of bytes equal to buffer_size.Slave will withhold the ack for the last byte if ack_last_byte field in protocol specific header is not set or else it will ack the last byte immediately.

SW should read the bytes received from SLV_RX_FIFO register.This will generate a trigger if the fifo is above threshold. Please see FIFO_CONTROL register on how to program threshold levels.Last two steps will be repeated every time there is a transfer equal to buffer_size.

To work in Tegra 3 packet mode for Rd operation(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 an rd transaction is received, 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, interrupt will again be generated and TX_BUFFER_REQ will be set.

The data can be sent/received without working in packet mode by only updating the following register field information

For ex, WR 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 HW needs to withhold ack for the last byte.
- After receiving SLV_RX_BUFFER_FILLED interrupt , clear the interrupt and update ACK_LAST_BYTE field. Also read the bytes from RX_FIFO

33.5 I2C Registers

33.5.1 I2C_I2C_CNFG_0

IC Controller Configuration Register (Master) 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 ACK.

Offset: 000h | Read/Write: R/W | Reset: 0b0000100000000000

Bit	Reset	Description
15	0x0	MSTR_CLR_BUS_ON_TIMEOUT: When this bit is set,I2C master will force clock lowfor an extended period of time(>TIMEOUT) to force all SMB slavesto release the bus0 = Dont clear the bus on TLow:SEXT/TIMEOUT timeout1 = clear the bus on TLow:SEXT/TIMEOUT timeout
14:12	0x0	DE-BOUNCE_CNT: De-bounce period for sda and scl lines 0 = No De-bounce 1 = 2T 2 = 4T 3 = 6T etc where T is the period of the fix PLL clk source coming to i2c.Maximum De-bounce period programmable is 14T. A De-bounce period of >50ns is desirable
11	0x1	NEW_MASTER_FSM: Maintained for compatibility sake. 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. Hence, firmware should first configure all other registers and bits [8:0] of 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 ACK. 1 - don’t look for an

Bit	Reset	Description
		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 slave1. 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= 1byte ... 111 = 8bytes; In a two slave transaction number of bytes should be programmed 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

33.5.2 I2C_I2C_CMD_ADDR0_0

I2C_CMD_ADDR0 is programmed the 7 Bit or 10 Bit address of slave 1 with which the transaction is intended.

I2C Slave-1 Address

Offset: 004h | Read/Write: R/W | Reset: 0b0000000000

Bit	Reset	Description
9:0	0x0	ADDR0: In case of 7-Bit mode address is written in the I2C_CMD_ADDR0[7:1] and I2C_CMD_ADDR0[0] indicates the read/write transaction.I2C_CMD_ADDR0[0] bit must match with the I2C_CNFG[6]. In case of 10-Bit mode address is written in I2C_CMD_ADDR0[9:0] and I2C_CNFG[6] indicates the read/write transaction.

33.5.3 I2C_I2C_CMD_ADDR1_0

I2C_CMD_ADDR1 is programmed the 7 Bit or 10 Bit address of slave 2 with which the transaction is intended.

I2C Slave-2 Address

Offset: 008h | Read/Write: R/W | Reset: 0b0000000000

Bit	Reset	Description
9:0	0x0	ADDR1: In case of 7-Bit mode address is written in the I2C_CMD_ADDR0[7:1] and I2C_CMD_ADDR0[0] indicates the read/write transaction.I2C_CMD_ADDR0[0] bit must match with the I2C_CNFG[7]. In case of 10-Bit mode address is written in I2C_CMD_ADDR0[9:0] and I2C_CNFG[7] indicates the read/write transaction.

33.5.4 I2C_I2C_CMD_DATA1_0

The four Least Significant Bytes of Data to be transmitted is loaded into the register when I2c Master is in Write Mode.

The four Least Significant Bytes of Data are Read through this register when I2C Master is in Read mode.

IC Controller Data 1: Transmit/Receive

Offset: 00ch | Read/Write: R/W | Reset: 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: This register contains the first data byte to be sent/received.

33.5.5 I2C_I2C_CMD_DATA2_0

The four Most Significant Bytes of Data to be transmitted is loaded into the register when I2C Master is in Write Mode.

The four Most Significant Bytes of Data are Read through this register when I2C Master is in Read mode.

IC Controller Data 2: Transmit/Receive

Offset: 010h | Read/Write: R/W | Reset: 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: This register contains the Fifth data byte to be sent/received.

33.5.6 I2C_I2C_STATUS_0

I2C_STATUS gives the status of I2C master operation.

IC Controller Status (Master)

Offset: 01ch | Read/Write: RO | Reset: 0bxxxxxxxxx

Bit	Reset	Description
8	X	BUSY: 1 = Busy. 0 = NOT_BUSY 1 = BUSY
7:4	X	CMD2_STAT: Transaction for Slave2 for x byte failed. x is 'h0 to 'ha. all others 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
3:0	X	CMD1_STAT: Transaction for Slave1 for x byte failed. x is 'h0 to 'ha. all others invalid 0 = SL1_XFER_SUCCESSFUL 1 = SL1_NOACK_FOR_BYTE1 2 = SL1_NOACK_FOR_BYTE2 3 = SL1_NOACK_FOR_BYTE3

Bit	Reset	Description
		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

33.5.7 I2C_I2C_SL_CNFG_0

I2C_SL_CNFG register is used to configure,

- Enable mode of slave Ack
- Enable mode of slave response to general call address

The register should be programmed when I2C controller is configured as slave.

IC Controller Configuration (Slave)

Offset: 020h | Read/Write: R/W | Reset: 0b00000000000000000100

Bit	Reset	Description
20	0x0	FIFO_XFER_EN: If this bit is disabled, data is always communicated via I2C_SL_RCVD register. If enabled, it is through FIFOs 0 = DISABLE 1 = ENABLE
19:8	0x0	BUFFER_SIZE: Payload size in bytes
7	0x0	ACK_LAST_BYTE_VALID: ack the last byte valid(Write-Only) This bit qualifies ACK_LAST_BYTE field 0 = DISABLE 1 = ENABLE
6	0x0	ACK_LAST_BYTE: ack 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,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 - slave will not ack reception of address or data byte. 0 = DISABLE 1 = ENABLE
0	0x0	RESP: Slave response to general call address (zero address) 1 - Enable. 0 = DISABLE 1 = ENABLE

33.5.8 I2C_I2C_SL_RCVD_0

IC Controller Slave Receive/Transmit Data (Slave)

Offset: 024h | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	SL_DATA: slave Received data

33.5.9 I2C_I2C_SL_STATUS_0

IC Controller Slave Status (Slave)

Offset: 028h | Read/Write: R/W | Reset: 0bxxxxxxx000000x0

Bit	R/W	Reset	Description
14:8	RO	X	HW_MSTR_ADR: HW master addr 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. 1 = Received HW Master Address 0 = No event.
6	RW	0x0	REPROG_SL: 1 = Interrupt has been generated by slave after General Call Address is 0x04. 1 = Reprogram slave address. 0 = No action.
5	RW	0x0	RST_SL: 1 = Interrupt has been generated by slave after General Call Address is 0x06. 1 = 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. 1 = Transaction completed. 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 = 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

33.5.10 I2C_I2C_SL_ADDR1_0

IC Controller Slave Address 1 Register (Slave)

Offset: 02ch | Read/Write: R/W | Reset: 0b0000000000000000

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.

33.5.11 I2C_I2C_SL_ADDR2_0

IC Controller Slave Address 2 Register (Slave)

Offset: 030h | Read/Write: R/W | Reset: 0b0xxxxx000xxxxx000

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 MSB of the address.
0	0x0	VLD: 0 = 7-bit addressing. 1 - 10 bit addressing. 0 = SEVEN_BIT_ADDR_MODE 1 = TEN_BIT_ADDR_MODE

33.5.12 I2C_I2C_TLOW_SEXT_0

IC Controller SMBUS timeout thresholds

Offset: 034h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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 timeout period in milliseconds

33.5.13 I2C_I2C_SL_DELAY_COUNT_0

IC Slave Controller Delay Count

Offset: 03ch | Read/Write: R/W | Reset: 0b0000000000011110

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 internal slave is accessed. I2C_SL_DELAY_COUNT has to be programmed such that TIMING = T * DLY where T is period of clock source selected for I2c; and DLY is I2C_SL_DELAY_COUNT; TIMING is the desired timing, A value of >= 1250 ns is advisable.

33.5.14 I2C_I2C_SL_INT_MASK_0

IC Controller Slave Mask (Slave)

Offset: 040h | Read/Write: R/W | Reset: 0b111111x1

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

33.5.15 I2C_I2C_SL_INT_SOURCE_0

IC Controller Slave Source (Slave)

Offset: 044h | Read/Write: RO | Reset: 0bxxxxxxxx

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

Bit	Reset	Description
2	X	RCVD: 0 = UNSET 1 = SET
0	X	ZA: 0 = UNSET 1 = SET

33.5.16 I2C_I2C_SL_INT_SET_0

IC Controller Slave Source (Slave)

Offset: 048h | Read/Write: R/W | Reset: 0b000000x0

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

Imp Note: Program the field PACKET_MODE_EN of I2C_CNFG register while working in packet mode.

The set of registers below describe the interface for packet mode only. With the 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:

1. There is no restriction on the no. of bytes before and the after the repeated start
2. The number of bytes that can be transferred with a single cmd is not limited to 8. Though a packet can contain 4k bytes, because any number of packets can be pushed into the fifo, there is no limit on the no. of bytes that can be transferred.
3. The transactions to different slaves can be chained together with repeat start and there is no limit on the no. of slaves it can address.

33.5.17 I2C_I2C_TX_PACKET_FIFO_0

A packet contains header and payload. Header size is variable and could vary from 2 to 5 words. For I2C it is 3 words. The first two words of the header contain generic information. The third word contains I2C transaction specific information. Payload contains actual data to be written to the slave. In case of read operation, payload is nil, hence the packet contains header only.

Offset: 050h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_PACKET: SW writes packets into this register. A packet may contain generic

33.5.18 I2C_I2C_RX_FIFO_0

Header or I2C Specific Header or Data

Offset: 054h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	RD_DATA: SW Reads data from this register cause a pop

33.5.19 I2C_PACKET_TRANSFER_STATUS_0

Offset: 058h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
24	X	TRANSFER_COMPLETE: The packet transfer for which 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 addr 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

33.5.20 I2C_FIFO_CONTROL_0

Offset: 05ch | Read/Write: R/W | Reset: 0b0000000000000000

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 empty in the FIFO. 001 = 2 word, DMA trigger is asserted when at least 2 words empty in the FIFO
12:10	0x0	SLV_RX_FIFO_TRIG: Slave Receive FIFO trigger level. 000 = 1 word DMA trigger is asserted when at least one word full in the FIFO. 001 = 2 word DMA trigger is asserted when at least 2 word full in the FIFO
9	0x0	SLV_TX_FIFO_FLUSH: 1= flush the tx FIFO, cleared after FIFO is flushed 0 = UNSET 1 = SET

Bit	Reset	Description
8	0x0	SLV_RX_FIFO_FLUSH: 1= flush the rx FIFO,cleared after 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 empty in the FIFO 001 = 2 word, DMA trigger is asserted when at least 2 words 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 full in the FIFO 001 = 2 word DMA trigger is asserted when at least 2 word full in the FIFO
1	0x0	TX_FIFO_FLUSH: 1= flush the TX FIFO, cleared after FIFO is flushed 0 = UNSET 1 = SET
0	0x0	RX_FIFO_FLUSH: 1= flush the RX FIFO, Cleared after FIFO is flushed 0 = UNSET 1 = SET

33.5.21 I2C_FIFO_STATUS_0

Offset: 060h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

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 = Master terminated transaction before it was completed 1 = Master did not terminate 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 empty0001 = 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

33.5.22 I2C_INTERRUPT_MASK_REGISTER_0

Offset: 064h | Read/Write: R/W | Reset: 0b000000000xx00xxxxx000x0000000

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

Bit	Reset	Description
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
10	0x0	TLOW_MEXT_TIMEOUT_EN: 0 = DISABLE 1 = ENABLE
9	0x0	TLOW_SEXT_TIMEOUT_EN: 0 = DISABLE 1 = ENABLE
8	0x0	TIMEOUT_INT_EN: 0 = DISABLE 1 = ENABLE
6	0x0	ALL_PACKETS_XFER_COMPLETE_INT_EN: 0 = DISABLE 1 = ENABLE
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

33.5.23 I2C_INTERRUPT_STATUS_REGISTER_0

This register indicates the status bit for which the interrupt is set. If set, Write 1 to clear it However TFIFO_DATA_REQ,RFIFO_DATA_REQ fields depend on the fifo trigger levels and cannot be cleared.

Slv_rd2wr indicates there is a switch from RD to WR by repeat start and the current RD transaction needs to be closed and started with WR transaction. Similarly slv_wr2rd indicates switch from WR to RD.

Offset: 068h | Read/Write: R/W | Reset: 0b000000000xxxxxxxxx000000000xx

Bit	R/W	Reset	Description
28	RW	0x0	SLV_ACK_WITHHELD: ack is withheld, waiting for software explicit information about ack 0 = UNSET 1 = SET
27	RW	0x0	SLV_RD2WR: Transaction switching from rd to wr 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
26	RW	0x0	SLV_WR2RD: Transaction switching from wr to rd 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
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 req 0 = UNSET 1 = SET
16	RO	X	SLV_RFIFO_DATA_REQ: slave Rx FIFO data req 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. TRANSFER_PKT_ID filed 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 the 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 req 0 = UNSET 1 = SET

Bit	R/W	Reset	Description
0	RO	X	RFIFO_DATA_REQ: RX FIFO data req 0 = UNSET 1 = SET

33.5.24 I2C_I2C_CLK_DIVISOR_REGISTER_0

The divisor value (N) must be programmed so that desired scl freq = Clk source freq/12*N. Typically clk source frequency is fixed at 72 MHz.

Offset: 06ch | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	I2C_CLK_DIVISOR_HSMODE: N= divide by n+1

33.5.25 I2C_I2C_INTERRUPT_SOURCE_REGISTER_0

This is a read-only register which returns the AND of Interrupt Source and Interrupt Mask registers.

Offset: 070h | Read/Write: RO | Reset: 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 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 req 0 = UNSET 1 = SET
16	X	SLV_RFIFO_DATA_REQ: slave Rx FIFO data req 0 = UNSET 1 = SET
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

Bit	Reset	Description
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 req 0 = UNSET 1 = SET
0	X	RFIFO_DATA_REQ: Rx FIFO data req 0 = UNSET 1 = SET

33.5.26 I2C_I2C_INTERRUPT_SET_REGISTER_0

This is a write-only register which 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. Read always returns 'h0.

Offset: 074h | Read/Write: R/W | Reset: 0b000000000xxxxxxxxx000000000

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
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

Bit	Reset	Description
20	0x0	SLV_RFIFO_UNF: slave Rx FIFO underflow 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

33.5.27 I2C_I2C_SLV_TX_PACKET_FIFO_0

Offset: 078h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TX_PACKET: SW writes packets into this register. A packet may contain generic

33.5.28 I2C_I2C_SLV_RX_FIFO_0

Header or I2C specific header or data

Offset: 07ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	0x0	RD_DATA: SW Reads data from this register cause a pop

33.5.29 I2C_I2C_SLV_PACKET_STATUS_0

Offset: 080h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
25	X	ACK_WITHHELD: Indicates that ack is withheld for last byte and slave is waiting for host to explicitly command slave to ACK the last byte0 = Bus is released1 = 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

[THIS PAGE INTENTIONALLY LEFT BLANK]

34.0 UART AND VFIR CONTROLLER

There are five Universal Asynchronous Receiver/Transmitters (UARTs) built into Tegra® 3 devices. These UARTs support both 16450 and 16550 compatible modes. All UARTs are identical, though their pin connections will be different. One of the UARTs also contains VFIR functionality.

All UARTs provides 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 and is selected by means of the FIFO control register. It also includes a 16-bit programmable baud rate generator and an 8-bit scratch register, eight modem control lines, and two 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, and is therefore 4.5 Megabits per second.

Note: Mention of UART A/B/C/D/E in this document equates to UART 1/2/3/4/5, respectively.

The Tegra 3 device VFIR controller 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.

Table 88 Protocols Supported by the IrDA

Signaling Rate (bps)	IrDA Name	Frame Wrapper and Encoding	IrDA Status	Tegra 3 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
16.0 M	VFIR	Sync HHH(1,13)	Optional	Yes

34.1 Hardware Features

- Synchronization for the serial data stream with start and stop bits to transmit data to and from a data character
- Data integrity by attaching parity bit to the data character
- The COM-HOST interface is fully programmable through an 8-bit CPU interface
- Support for word lengths from five to eight bits, an optional parity bit and one or two stop bits

- The UART supports both 16450- and 16550-compatible modes. Default mode is 16450
- DMA capable for both TX and RX
- 8 bit x 16 deep FIFO
- Auto sense baud detection

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
- Maximum frequency of device clock is 72 MHz

34.2 Hardware Signaling

All Tegra UARTs are implemented by a HW block that supports modem control signals such as DTR, DSR, DCD.

UART A is, however, the only UART that allows those signals to be used externally. For UART A, when the pinmux 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 HW. This means that if SW changes DTR state, the UART may raise an interrupt due to DSR/DCD changing state.

For UARTs C, D, and E, the DSR and DCD inputs are tied to an inactive value.

Table 89 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.

34.3 Functionality

The controller can operate in three different encodings and 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 3 device processor. The input signals are de-multiplexed in a similar fashion.

Note: The operating mode is selected with the Mode [2:0] bits of the VFIR.CTL register

Figure 87 UART Block Diagram with SIR Decoder and Encoder

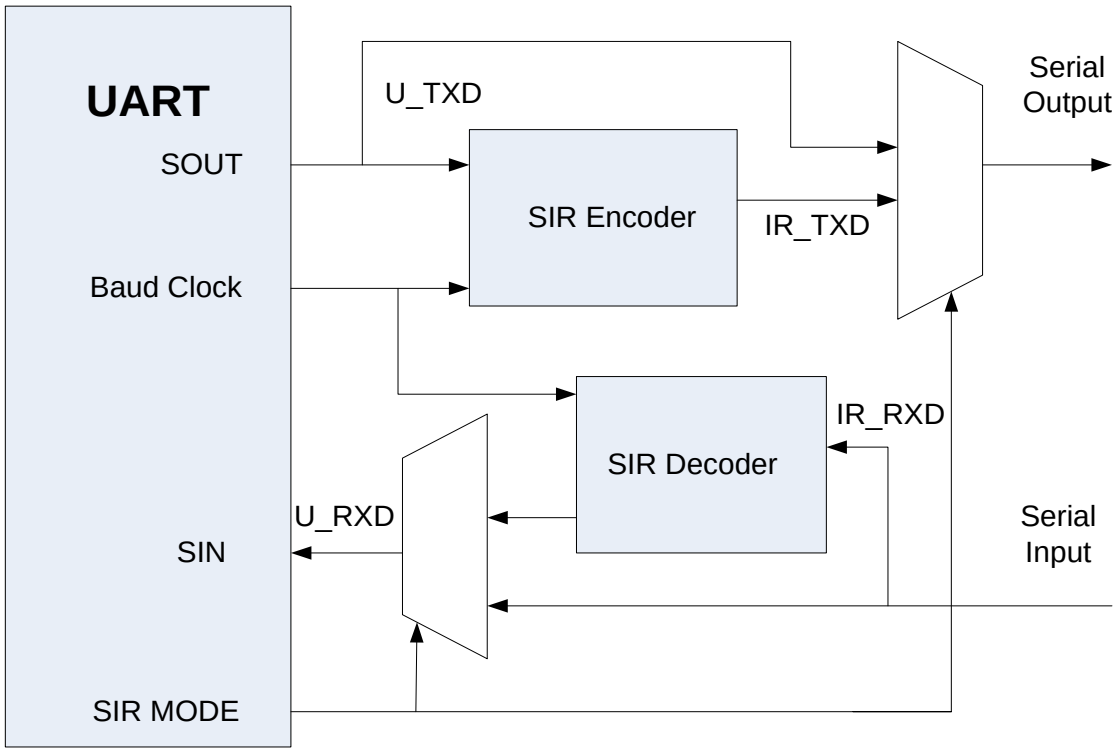
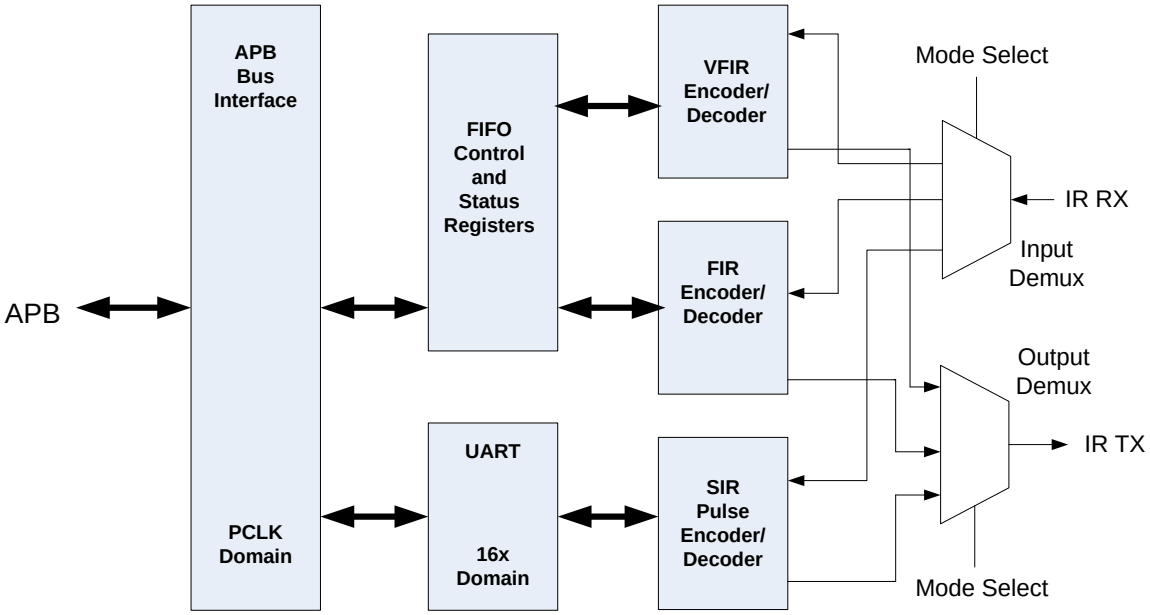


Figure 88 VFIR Controller Top-level Block Diagram



34.4 UART Programming Guidelines

34.4.1 16450 Mode Programming

1. Program pin mux settings to select UART
2. Enable UART clocks
3. For enabling internal loop-back, program MCR[4] to 1
4. Program FCR[0] to 0
5. Enable interrupts in IER register as needed
6. Write data into THR
7. Wait for THR interrupt if Interrupt is enabled or poll for LSR[5]
8. During receive, wait for RBR interrupt or poll for LSR[0]
9. Read the UART.LSR register to clear interrupts

34.4.2 16550 Mode Programming

1. Program pin mux settings to select UART
2. Enable UART clocks
3. For enabling internal loop-back, program MCR[4] to 1
4. Program FCR[0] to 1
5. Program trigger levels as required
6. Enable interrupts in IER register as needed
7. Write data into THR
8. Wait for THR interrupt if Interrupt is enabled or poll for LSR[5]
9. During receive wait for RBR interrupt or poll for LSR[0]
10. Read the UART.LSR register to clear interrupts
11. APB-DMA requestor number for UARTA, B and C are 8, 9 and 10 respectively

34.4.3 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.

34.4.4 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.

34.4.5 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.

34.4.6 Programmable Baud Rate Generator

The UART contains a programmable baud rate generator that is capable of taking the UART clock input and dividing it by any divisor from 1 to 2^16-1. The UART clocks are defined in section 6.1 on Clocking. 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.

34.4.7 Enable Register (IER)

The 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.

34.4.8 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, 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 90 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)

34.4.9 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.

34.4.9.1 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.

34.4.9.2 Receive operation in DMA mode 0

When 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.

34.4.9.3 Receive operation in DMA mode 1

When 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.

34.4.10 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.

34.4.11 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 91 Modem Control Register (MCR)

Bit	Name	Description
7:6	N/A	Not used. Set to 0 internally.
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

34.4.12 Line Status Register (LSR)

The Line Status Register provides the status of data transfer to the CPU.

34.4.13 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.

34.4.14 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.

34.4.15 Autobaud Sense Register (ASR)

The UART has the ability to 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 and when another rx_edge occurs, the "complete" flag is set, the value frozen and the autobaud sense value register updated with the count value. The low 20 bits of the ASR gives the number of clocks within a single bit. Because the UART uses 16x over sampling, 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 truewait4valid
```

```
        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
```

34.4.16 Baud Rate Generator

The following table given below is a divisor table for the baud rate, assuming the oscillator is 24.000 MHz.

Table 92 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

34.4.17 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

34.5 VFIR Programming Guidelines

34.5.1 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 modes. 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.

34.5.2 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.

34.5.3 FIR 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 100 (FIR)
4. FIR mode is Half-Duplex. Set the VFIR.CTL Transmit bit to '1' for transmit, '0' for receive
5. Read the VFIR.STS register to clear interrupts
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 interrupt which indicates that the transaction has completed
11. Check for Errors when reading the VFIR.STS register
12. Clear status bits by writing a '1' to them before starting 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.

34.5.4 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. FIR mode is Half-Duplex. Set the VFIR.CTL Transmit bit to '1' for transmit, '0' for receive
5. Read the VFIR.STS register to clear interrupts
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 interrupt which indicates that the transaction has completed
11. Check for Errors when reading the VFIR.STS register
12. Clear status bits by writing a '1' to them before starting 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.

34.5.5 COM-SLAVE Interface (UART B)

The COM-SLAVE interface consists of a UART (Universal Asynchronous Receiver/Transmitter), which provides serial data synchronization, 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 five to eight bits, an optional parity bit and one or two stop bits. If enabled, the parity can be odd, even or forced to a defined state. Interrupts can be generated from any of ten sources.

The UART supports both 16450 and 16550 compatible modes. Default mode is the 16450. The 16550 mode, which provides independent 16-byte FIFOs for transmit and receive, is selected via the FIFO control register. It also includes a 16-bit programmable baud rate generator and an 8-bit Scratch register, eight 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 93 Serial Bus Interface Signals for UART B

Signal Name	Direction	Description
UB3_RXD	I	Receiver Data Port
UB3_TXD	O	Transmit Data Port

34.5.6 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.

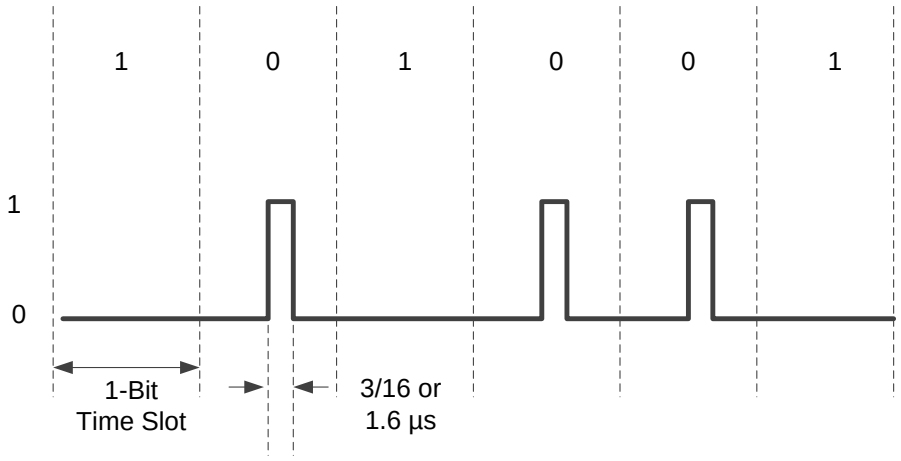
Similarly, received (RX) data is bit-synchronized using the 16X baud clock and the original serial data recovered from the IR bit stream. This data is then sent to the UART for reception.

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 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.

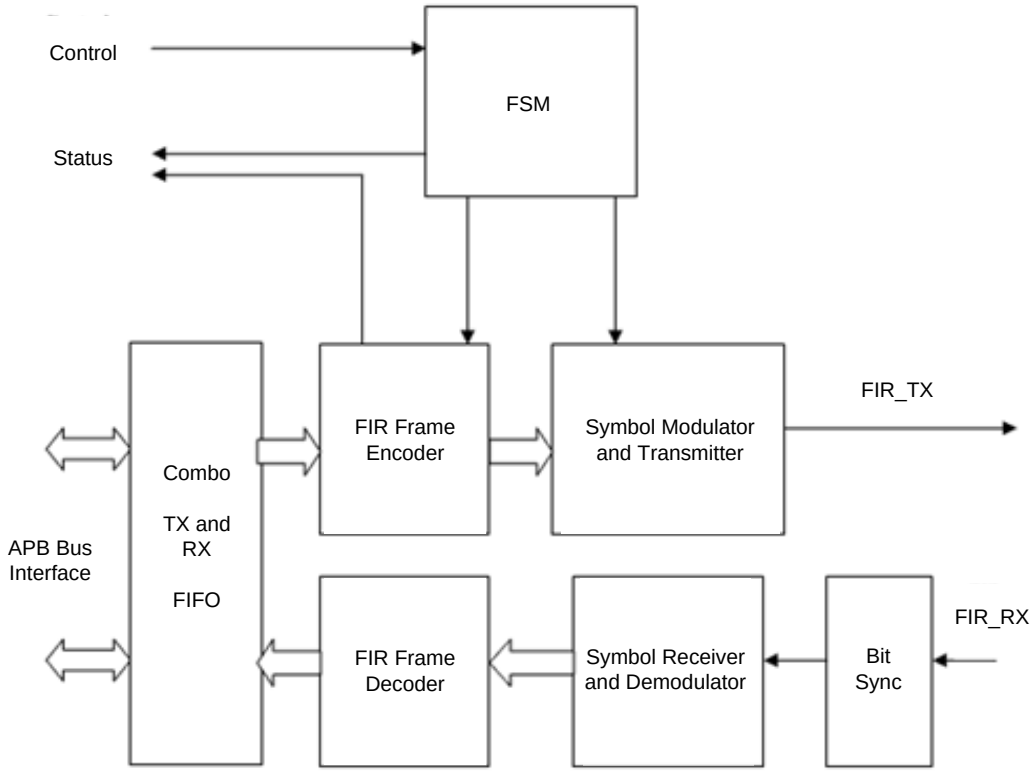
Figure 89 Output in Sending Order in SIR Encoding



34.5.7 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 90 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, 2-bit signals in FIR, 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.

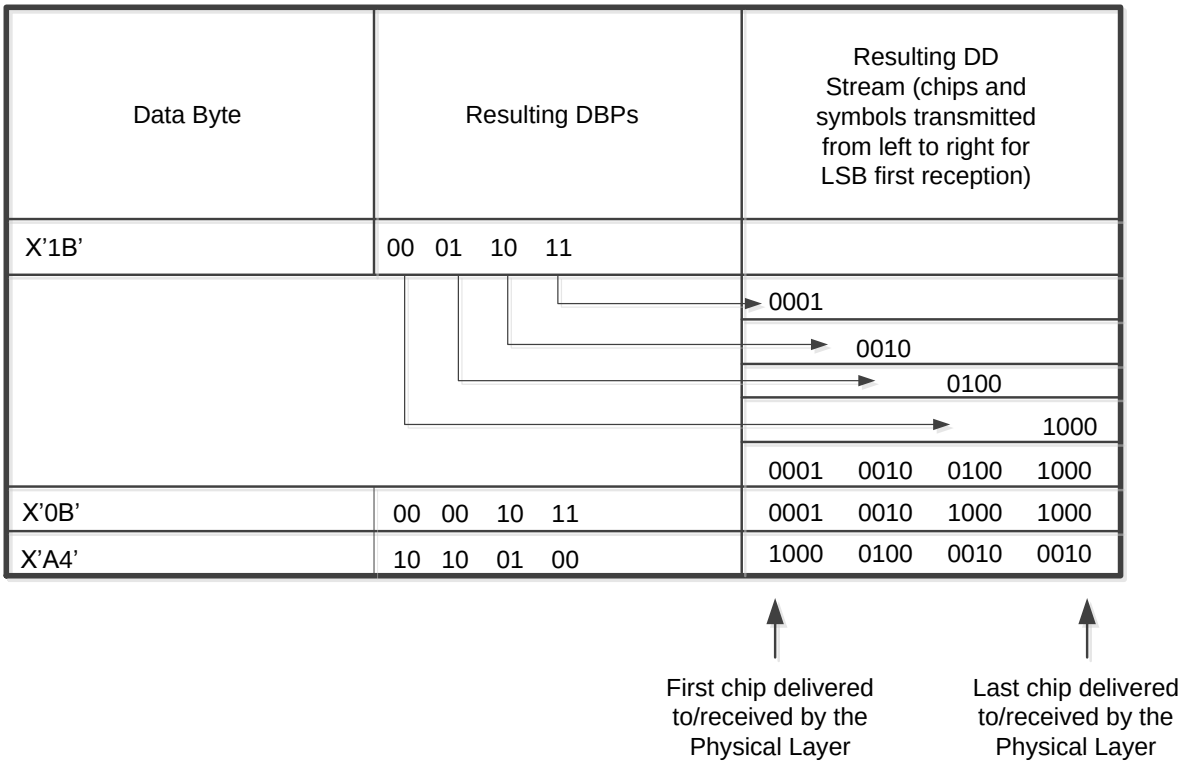
Table 94 Chip Pattern

Data Bit Pair	4PPM Data Symbol
00	1000
01	0100
10	0010
11	0001

Logical "1" represents a chip duration when the transmitting LED is emitting light, while 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 91 Various Data Bytes After Encoding for Transmission



34.5.8 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 following packet format is defined:

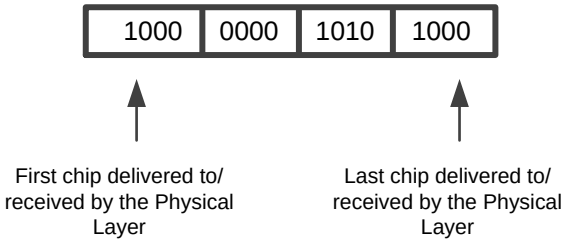
In this packet format, the payload data is encoded as described in the 4PPM encoding above, and the encoded symbols reside in the DD field. Maximum packet length is negotiated by the same mechanism as for the slower rates. The preamble field (PA) is used by the receiver to establish phase lock. During 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 (note that, 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 other4 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.

34.5.8.1 Preamble Field Definition

The preamble field (PA) consists of exactly sixteen 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.

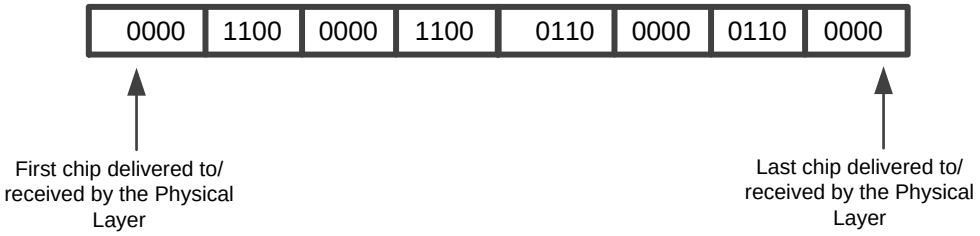
Figure 92 Preamble Field



34.5.8.2 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.

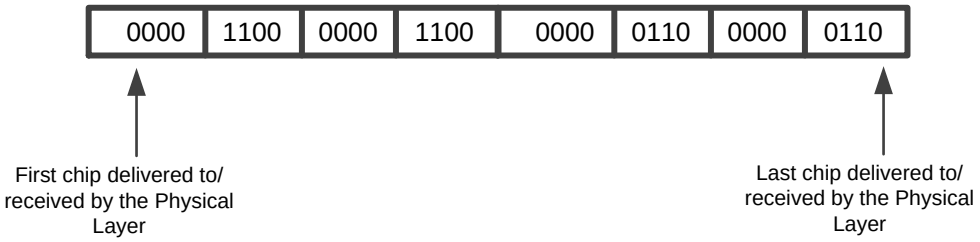
Figure 93 Start Flag



34.5.8.3 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 94 Stop Flag



34.5.8.4 Frame Check Sequence Field Definition

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 4 PPM 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.

34.5.8.5 Aborted Packets

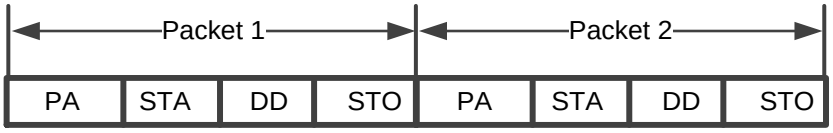
Receivers may only accept packets that have valid STA, DD, FCS, and STO fields as defined in the PPM Packet Format section. The PA field need not be valid in the received packet. All other packets are aborted and ignored.

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.

34.5.8.6 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 95 Back-to-Back Packet Transmission



34.5.9 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.

34.5.10 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 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.

34.5.10.1 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* 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 95: 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

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
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.

34.5.10.2 HHH (1, 13) Packet Format

The packet format for 16.0 Megabits per second HHH(1,13) has the following form:

PREAMBLE (PA)	START (STA)	IrLAP Frame	CRC	Flush Byte (FB)	STOP (STO)	NULL

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 (note that, 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.

34.5.10.3 Preamble Field Definition

The transmitted PREAMBLE (PA) is constructed by concatenating ten times (10) 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).

34.5.10.4 Start (STA) Flag Definition

The transmitted START (STA) delimiter is the 48-chip (2 $^{\mu}$s) 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.

34.5.10.5 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) |

34.5.10.6 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 s) sequence.

34.5.10.7 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 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.

34.5.10.8 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.

34.5.10.9 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.

34.5.10.10 Scrambling and Descrambling Functions

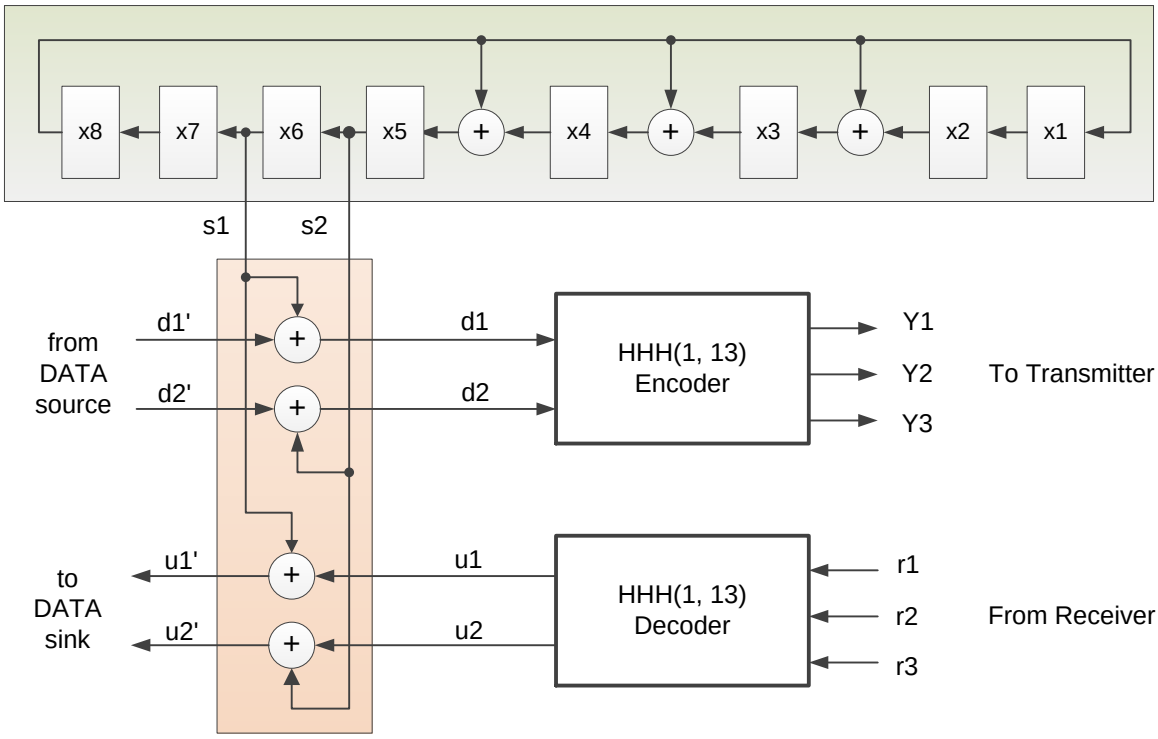
It is advantageous to enhance the encoder/decoder system with simple scrambler/descrambler functions.

The primitive polynomial: $x^8 \oplus x^4 \oplus x^3 \oplus x^2 \oplus 1$,

where ⊕ 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 96 Scrambling / Descrambling Scheme



34.5.10.11 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.

34.5.10.12 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.

34.5.10.13 Back to Back Packet Transmission

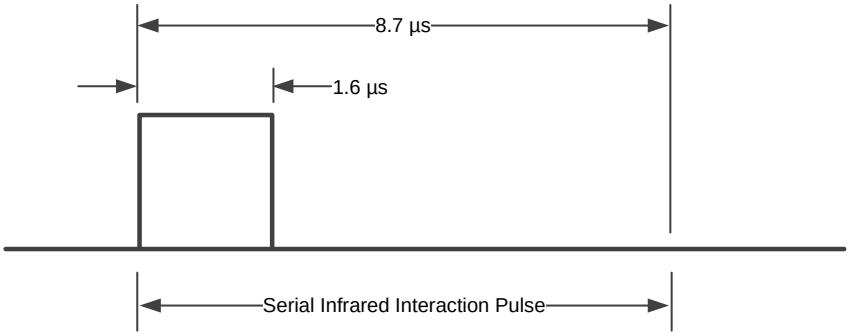
Back to back, or "brick-walled" packets are allowed, but each packet must be complete (i.e., containing PA, STA, IrLAP Frame, CRC and STO fields).

34.5.11 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.

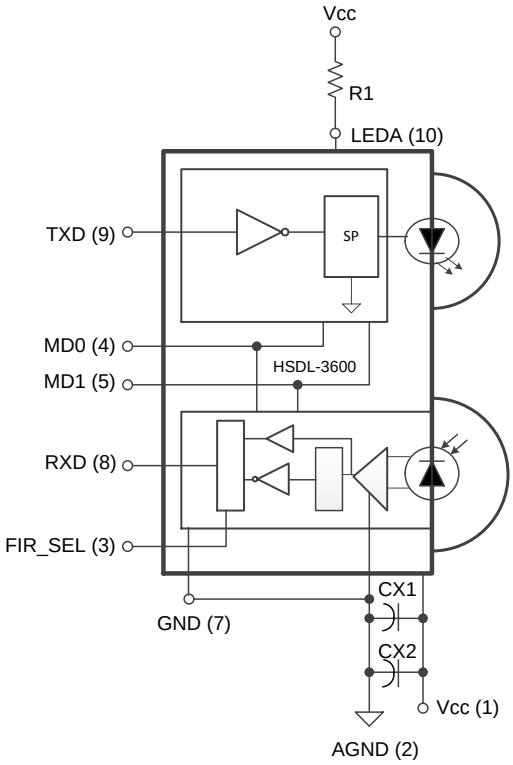
Figure 97 SIP



34.5.11.1 External Interface

The SIR/FIR/VFIR signals are muxed to 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 will all venders. 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 98 Typical Interface



34.6 UART Registers

34.6.1 UART_THR_DLAB_0_0

This 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.

UART Transmit Holding Register

Offset: 000h | Read/Write: R/W | Reset: 0b00000000

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)

34.6.2 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. 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: 004h | Read/Write: R/W | Reset: 0b000000

Bit	Reset	Description
5	0x0	IE_EORD: Interrupt Enable for End of Received Data 1 = Enable 0 = DISABLE 1 = ENABLE
4	0x0	IE_RX_TIMEOUT: Interrupt Enable for Rx FIFO timeout 1 = Enable 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

34.6.3 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 96 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: 008h | Read/Write: R/W | Reset: 0b00000000

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
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

Bit	Reset	Description
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 FIFO. This bit should be enabled 0 = DISABLE 1 = ENABLE

34.6.4 UART_LCR_0

UART Line Control Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000

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
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

Bit	Reset	Description
		1 = WORD_LENGTH_6 2 = WORD_LENGTH_7 3 = WORD_LENGTH_8

34.6.5 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 informs us that it is OK to send data.

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: 010h | Read/Write: R/W | Reset: 0b00000000

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 loopback of Serial Out to In 0 = DISABLE 1 = ENABLE
3	0x0	OUT2: nOUT2 (Not Used) 0 = DISABLE 1 = ENABLE
2	0x0	OUT1: nOUT1 (Not Used) 0 = DISABLE 1 = ENABLE
1	0x0	RTS: 0 = Force RTS to high if RTS HW 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

34.6.6 UART_LSR_0

UART Line Status Register

Offset: 014h | Read/Write: RO | Reset: 0bxxxxxxxxx

Bit	Reset	Description
8	X	TX_FIFO_FULL: Transmitter FIFO full status 0 = NOT_FULL 1 = FULL

Bit	Reset	Description
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

34.6.7 UART_MSR_0

UART Modem Status Register

Offset: 018h | Read/Write: R/W | Reset: 0b00000000

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

Bit	Reset	Description
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

34.6.8 UART_SPR_0

UART Scratch Pad Register

Offset: 01ch | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	SPR_A: Scratchpad register (not used internally)

34.6.9 UART_IRDA_CSR_0

Bits [7:6] control the InfraRed (SIR) encoding. If enabled, each zero bit sent will be 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: 020h | Read/Write: R/W | Reset: 0b00xx0000

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

34.6.10 UART_RX_FIFO_CFG_0

Offset: 024h | Read/Write: R/W | Reset: 0b0x000001

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 through 32)

34.6.11 UART_ASR_0

The UART has auto-Baud sensing logic which 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: 03ch | Read/Write: R/W | Reset: 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.

34.7 VFIR Registers

34.7.1 VFIR_CTL_0

VFIR Control register is used

- to enable/disable entire IRDA module
- to configure the controller in FIR mode or VFIR mode or SIR or UART mode
- to configure either in Transmit mode or Receive mode to invert the polarity of Tx or Rx line
- to enable/disable DMA
- to know the attention levels of Transmit FIFO and Receive FIFO
- to start Transmit or Receive operation
- to select the Frequency
- to clear Transmitter FIFO/Receiver FIFO

VFIR Control Register

Offset: 000h | Read/Write: R/W | Reset: 0b00000000000000000000000000100000

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 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 transmit. The 72MHz clock can be used in case the 48MHz 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 use UART B module 7 = RSVD 6 = RSVD1 5 = VFIR 4 = FIR 3 = RSVD2 2 = MIR 1 = SIR 0 = UARTB

34.7.2 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 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 in Transmitting/Receiving the data
- Whether the VFIR interrupt is generated or not. To generate interrupt the corresponding the enable bits has to be set in the VFIR Interrupt Enable Register

VFIR Status and Interrupt Identification Register

Offset: 004h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxxxxx0xxx0x000000

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 Tx FIFO
23	RO	X	RX_FTRIG: Receiver FIFO Trigger level Reached

Bit	R/W	Reset	Description
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 Rx FIFO
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 bit stream.
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.

34.7.3 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: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

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

34.7.4 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: 00ch | Read/Write: R/W | Reset: 0b0000000000000000

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.

34.7.5 VFIR_IFDL_0

VFIR Incoming Frame Data Length gives the information of the data received in bytes.

VFIR Incoming Frame Data Length

Offset: 010h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxx

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.

34.7.6 VFIR_FIFO_0

VFIR FIFO is a shared FIFO. Holds the data in Transmitter/Receiver mode

VFIR.FIFO (VFIR FIFO)

Offset: 040h | Read/Write: R/W | Reset: 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.

[THIS PAGE INTENTIONALLY LEFT BLANK]

35.0 SLINK: SPI PERIPHERAL INTERFACE

The SPI bus is a master-slave based bus that is used to interconnect different devices to a central master.

It allows full-duplex synchronous-serial communication between the controller and external peripheral devices. It consists of 4 signals, SS_N (Chip select), SCK (clock), MOSI (Master data out and Slave data in) and MISO (Slave data out and master data in). The data is transferred on MOSI or MISO based on the data transfer direction on every SCK edge. The receiver always receives the data on the other edge of SCK.

The clock polarity and clock phase determine which edge of the SCK clock will be used to latch the data in the receiver. If the clock polarity is 0, then the SCK signal is 0 when idle and transitions to 1 whenever doing a data transfer. If the clock polarity is 1, then the SCK signal is 1 when idle and transitions to 0 when doing a data transfer. If the clock phase is 0, then the receiver latches the data on the first transition of SCK from idle state. If the clock phase is 1, then the receiver latches the data on the second transition of SCK.

This makes up 4 modes for SPI protocol that are described below:

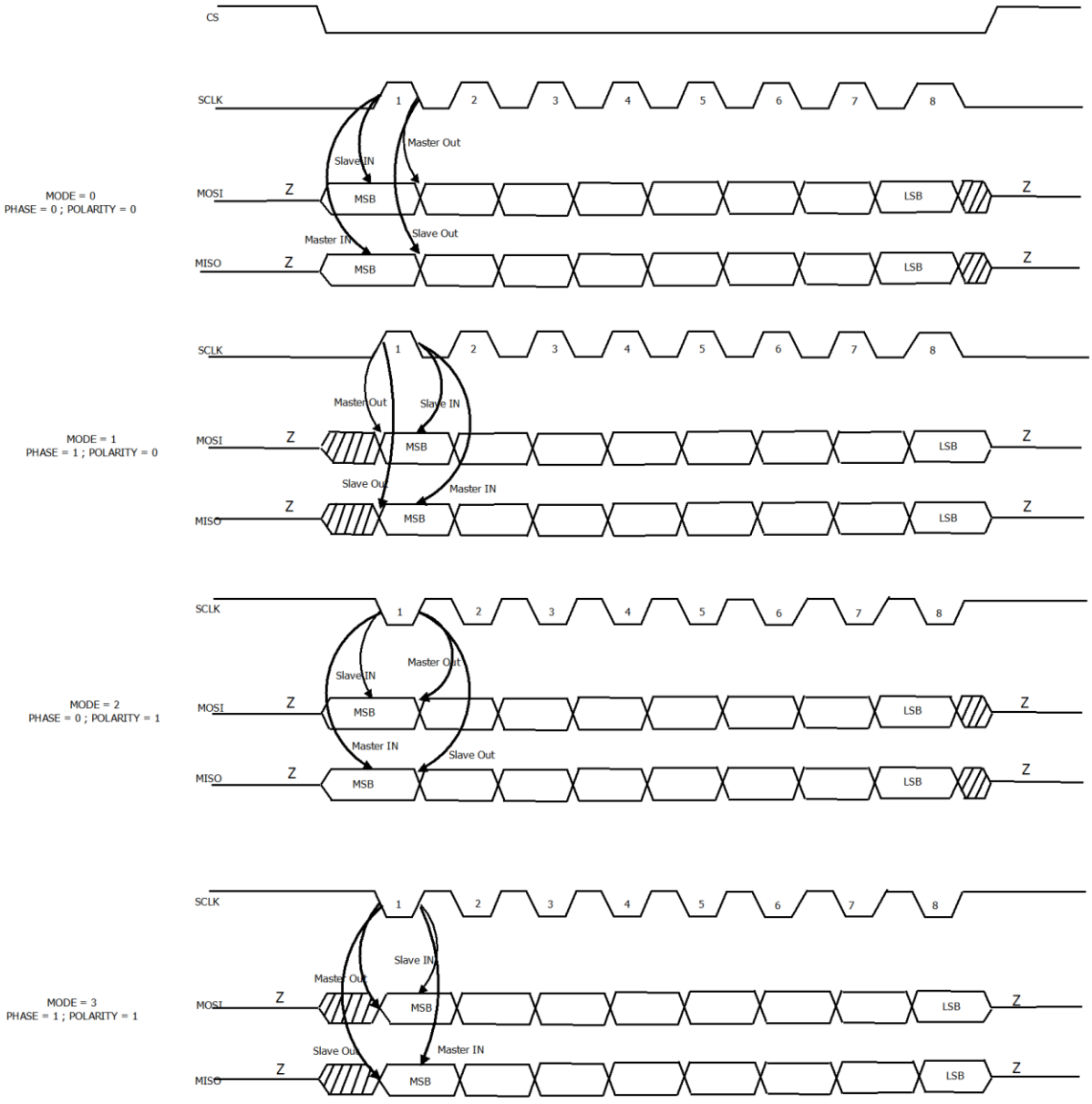
Table 97. SPI Protocols

SPI Mode	Clock Polarity	Clock Phase	Description
0	0	0	Clock is positive polarity and the data is latched on the positive edge of SCK.
1	0	1	Clock is positive polarity and the data is latched on the negative edge of SCK.
2	1	0	Clock is negative polarity and the data is latched on the negative edge of SCK.
3	1	1	Clock is negative polarity and the data is latched on the positive edge of SCK.

In this protocol, data is transferred on every edge of SCK whenever CS is active. CS can be active high or low. The data transfer consists of packets of varying bit lengths from 1 to 32 bits in any direction. Within a packet, the transfer starts with MSB first. The clock can stay in idle state in between different packets. CS can be active for either single packet or multiple packets including transmit and receive.

This protocol is shown in the diagram below.

Figure 99. SPI Interface Timing



Features

- Independent RX FIFO and TX FIFO.
- FIFO depth of 32x32 bits.
- Software controlled bit-length from 0 to 31 resulting in packet sizes of 1 to 32 bits.
- Packed mode support for bit-length of 7 (8-bit packet size) and 15 (16-bit packet size).
- SS_N can be selected to be controlled by software, or it can be generated automatically by the hardware on packet boundaries.
- Receive compare mode where the controller listens for a particular pattern on the incoming data before receiving the data in the FIFO.
- Simultaneous receive and transmit supported.
- Maximum transfer rate is 50 Mbits/sec, subject to board electrical characteristics.

35.1 Programming Guidelines

There are two basic modes of operation: DMA_EN and GO modes. It is required that software sets up all the parameters in the SBCx_COMMAND and SBCx_COMMAND2 registers before enabling transfers in any of these modes.

35.1.1 DMA_EN Mode

This mode is enabled by writing 1 to DMA_EN bit in SBCx_COMMAND register. In this mode, SPI controller transmits or receives the number of packets as indicated by the field DMA_BLOCK_SIZE in SBCx_DMA_CTL register

If PACKED bit is set and the BIT_LENGTH is set to 7, then all FIFO words contain 4 packets to transfer (transmit or receive). Packets will be transferred in little-endian format, with packet 0 being in byte 0 of the FIFO and packet 3 in byte 3 of the FIFO.

If PACKED bit is set and the BIT_LENGTH is set to 15, then all FIFO words contain 2 packets to transfer (transmit or receive). Packets will be transferred in little-endian format, with packet 0 being in bits [15-0] of the FIFO and packet 1 in bits [31-16] of the FIFO.

If BIT_LENGTH is set to N, each packet will consist of N + 1 bits. These bits will be transmitted/received in the TX_FIFO/RX_FIFO with the MSB in bit N and LSB in bit 0, following little-endian architecture. Any remaining bits in the FIFO will be ignored by the hardware. Maximum packet length is 32, which can be selected by setting BIT_LENGTH to 31. In this case, all data bits in FIFO contain valid packet data.

DMA request will be generated to APB_DMA in this mode depending on the setting of IE.TXC and IE.RXC. If transmit is enabled, setting IE.TXC to 00 will generate a TX DMA request whenever the TX FIFO has one word of space available (is not full). Setting IE.TXC to 01 will generate a TX DMA request whenever the TX FIFO has 4 words of space available (is empty). If receive is enabled, setting IE.RXC to 00 will generate a RX DMA request whenever the RX FIFO has one word of data available (is not empty). Setting IE.RXC to 01 will generate a RX DMA request whenever the RX FIFO has 4 words of data available (is full).

The APB DMA requester numbers for the SPI controllers are 15, 16, 17, 18, 27, and 28.

35.1.2 GO Mode

> **Note:** GO Mode is NOT supported in Full Duplex mode of operation. Full duplex is simultaneous transmit and receive.

This mode is enabled by writing 1 to GO bit in SBCX_COMMAND register. In this mode, if transmit is enabled, then the SPI controller transmits the data in the TX FIFO until TX FIFO is empty. If receive is enabled, then the SPI controller receives one packet of data in the RX FIFO.

Packed mode is supported in transmit only. If PACKED bit is set and the BIT_LENGTH is set to 7, then all TX FIFO words contain 4 packets to transfer. If PACKED bit is set and the BIT_LENGTH is set to 15, then all TX FIFO words contain 2 packets to transfer.

35.1.3 Interrupt Generation

SPI controller generates an interrupt to processor at the end of a transfer or when an error is detected if IE.TXC or IE.RXC bit in SBCX_DMA_CTL register is enabled for transmit and receive modes of operation respectively. This functionality is common to both DMA_EN and GO modes.

If IE.TXC is enabled during transmit, an interrupt is generated whenever RDY or TXF_OVF bit in SBCX_STATUS register is set to 1. During transmit, if software/APB DMA cannot fill the transmit FIFO fast enough, hardware waits for the software to fill up the TX FIFO and an under run condition is never generated. However, if software tries to write to a full TX FIFO, hardware sets TXF_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.RXC is enabled during receive, an interrupt is generated whenever RDY or RXF_UNR bit in SBCX_STATUS register is set to 1. During receive, if software/APB DMA cannot read the receive FIFO fast enough, hardware waits for the software to read out the RX FIFO and an overflow condition is never generated. However, if software tries to read from an empty RX FIFO, hardware sets RXF_UNR bit as an indication that software attempted to under run the RX FIFO. Hardware makes sure that the read actually never goes to an empty RX FIFO.

The interrupt can be cleared by clearing the source of the interrupt. If the interrupt is generated by assertion of RDY, then writing a 1 to RDY bit clears the interrupt. If the interrupt is generated by assertion of TXF_OVF, then writing a 1 to TXF_OVF bit clears the interrupt. If the interrupt is generated by assertion of RXF_UNR, then writing a 1 to RXF_UNR bit clears the interrupt.

SPI Controller interrupt source is bit SPI (bit 7) in secondary interrupt controller.

35.1.4 Packet Parameters

Clock signal (SCK) output format: The SCK clock signal output format is controlled by the field IDLE.SCLK in SBCX_COMMAND register.

Clock Phase: The phase of the SCK signal is determined by the field CK.SDA in SBCx_COMMAND

Both clock polarity and clock phase together determine the SPI mode as described earlier.

Data signal (SDO) output format: The output format of the SDO signal is determined by IDLE.SDA field in SBCX_COMMAND register.

Transmit mode: Transmit mode is enabled by setting TXEN bit in SBCX_COMMAND register.

Receive mode: Receive mode is enabled by setting RXEN bit in SBCX_COMMAND register.

Chip select: The SPI CS signal can be controlled either by software or by hardware based on the value programmed in CS_SW bit in SBCX_COMMAND register.

If CS_SW bit is set to 1, the SPI CS works in software mode and any value programmed in CS bit in SBCX_COMMAND register appears on SPI CS. In this mode, software is responsible to generate appropriate chip select polarity by writing correct values in CS bit. This mode can be used to do data transfers that require multiple packets within a single chip select assertion.

If CS_SW bit is set to 0, the SPI CS works in hardware mode. In this mode, CS bit in SBCX_COMMAND register works as polarity of the SPI_CS signal. The SPI_CS will be driven active on a per-packet basis. It will be driven inactive in between packets and when the SPI bus is idle. In this mode, CS_DELAY determines the number of cycles the SPI_CS signal stays inactive in between two packets. Setting CS_DELAY to 0 gives a delay of 2 SPI_SCK cycles in between two packets if receive mode is enabled. Any other number N written to this field will give a delay of N + 2 cycles if receive mode is enabled. If only transmit mode is enabled, setting CS_DELAY to N will give a delay of N cycles.

35.1.5 Status Information

There are certain bits of status information in SBCX_STATUS register that software can use to determine the status of the currently ongoing transfer.

BSY bit: This bit is set to 1 at the start of every transfer and is cleared when the transfer has finished.

RDY bit: This bit is set to 1 at the end of every transfer. It is cleared when software writes a 1 to this bit.

RXF_UNR bit: This bit indicates an under run in RX FIFO and is set to 1 whenever an error is detected during a receive operation.

TXF_OVF bit: This bit indicates an overflow in TX FIFO and is set to 1 whenever an error is detected during a transmit 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 and BSY is 1, 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.

TXF_FULL: This bit indicates whether the TX FIFO is full or not and is set to 1 when TX FIFO is full.

TXF_EMPTY: This bit indicates whether the TX FIFO is empty or not and is set to 1 when TX FIFO is empty. If this bit is set to 1 during a transmission and BSY is 1, it indicates that the controller has transmitted all data from the TX FIFO and is waiting for firmware/APB DMA to write more data before continuing with the transmission.

CUR_BLOCK_COUNT: This field contains the no. of packets that have been transferred (sent or received) during current transfer.

If both transmit and receive are enabled and either RXF_FULL or TXF_EMPTY is set, controller will pause both the transmission and reception regardless of whether RXF_FULL or TXF_EMPTY is set.

35.1.6 Clock Initialization and Control

The SPI controller is targeted to run at 25 MHz at its interface to external SPI devices. The internal SPI controller clock should then run at 4X of the external SPI interface clock. Hence, the internal SPI controller clock sources for SPI 1, SPI 2, and SPI 3 all need to run at 100 MHz in this case. This 4X ratio is required for the correct operation of the SPI controllers.

The CLK_OUT_ENB_H register in the CAR (Clock and Reset) contains the enable bits for the 3 instances of the SPI controller – SPI 1 (bit 9 - CLK_ENB_SBC1), SPI 2 (bit 12 - CLK_ENB_SBC2) and SPI 3 (bit 14 - CLK_ENB_SBC3).

In addition, the clock source and divider registers CLK_SOURCE_SBC1, CLK_SOURCE_SBC2 and CLK_SOURCE_SBC3 registers contain the 2-bit field SBCx_CLK_SRC to decide the PLL clock source while the field SBCx_CLK_DIVISOR determines the divide ratio.

35.1.7 Programming Limitations

In packed transfers, you must follow a 3 step sequence.

- Program the bit length and packet parameters
- Enable packed mode
- Enable DMA

If the SPI interface is running at slow speeds such as at 1 MHz then there is an additional delay up to 1us required between enabling Packed mode and enabling DMA.

- Programming of both the bit lengths and pack_size. When in packed mode, the controller should not depend on bit-length. Alternatively, the controller can rely on bit_length if the controller is not in packed mode.
- In packed mode the number of blocks transferred need to be aligned by boundary. i.e. for a packed 4 the number of packets should be multiple of 8 and for packed 8 it has to be multiple of 4.
- With SPI clock 4 times faster than APB clock, there is a restriction on the bit-length. If the number of bits are <4, then the busy goes low as soon as the words are received on the SPI clock. But it doesn't wait till these are written into FIFO. And since APB clock is running slow, there is a possibility of data lost if less number of bits are transferred.
- Busy bit does not go low if the transfer is terminated in the middle. Hence termination in the middle of a transfer is not recommended.
- DMA should be programmed after APB is programmed in APB-DMA transfers.
  - If more than 32 words need to be transferred, then wait for the FIFO to be full before you start the SPI controller to transmit
  - 32 or less words need to be transferred, then first fill up the FIFO before the SPI controller is started to transmit

- Full duplex transfer in GO mode - First word in RXFIFO is always 0x00. It is recommended to use the DMA mode instead.
- Programming of IDLE_SCK. The requirement is to always program the same value in IDLE_SCK.
- Similar is the case for IDLE_SDA as for IDLE_SCK.
- Implementation of WAIT for preventing writes to command register till the current transfer is finished.
- Use of master and slave buffers. As of now they don't do anything except storing the last word transmitted/received from/to TX/RX FIFO.
- SS_SETUP bit is reserved.
- If the SPI interface clock is run at very low frequencies (such as 1 MHz), it may be necessary to add delays in the programming sequence as specified in (1) above so that the programming can take effect.

35.2 SPI Controller as a Slave

The SPI controller can also be used as a slave. In this mode the controller receives the chip select and the clock from the external master. In addition, data input and outputs are swapped on the two data lines - MOSI and MISO. The table below provides additional information on the SPI signals in master and slave configurations:

Table 98. SPI Signals in Master and Slave Configurations

	Slave	Master
MOSI	Input	Output
MISO	Output	Input
SPI_CSx_N	Input	Output
SCK	Input	Output

SCK

The slave derives the clock from the external master. It is required that between two consecutive words (up to 32 bits), 1 cycle delay be provided for correct operation. This is mandatory for both the DMA and GO mode of operation. If the packed mode is selected (either one of 4, 8, 16 or 32 bits), a 1 cycle delay is required between two consecutive packets.

SS_N

The chip select will need to be asserted by the external master and will stay asserted during the entire duration of the operation. If the chip select is de-asserted during the middle of a transfer, the SPI controller will cease operation until the chip select is asserted. If the external master continues to provide clocks while the chip select stays de-asserted, it has no effect on the SPI controller. It is required that the chip select be asserted and the operation completed in full if such an event should occur.

MOSI and MISO

The data pins MOSI and MISO are now swapped while the SPI controller is operated as a slave. The SPI controller now drives out the data on MISO (instead of on MOSI when it is used as a master) and receives data on MOSI (as supposed to on MISO when it is used as a slave).

PIO and DMA Modes

For large transfers of data over the SPI interface, APB DMA could be used and the DMA mode should be selected in the SPI. This will enable the transfer of 2**16 packets (4, 8, 16, or 32 bits in each packet) over the SPI interface. The PIO mode can also be used to fill up the FIFO in the controller but may require higher latency to transfer such high data rates.

35.3 SLINK Registers

35.3.1 Programming Sequence for Basic Data Transmission

After the module-initialization (reset and clock programming), program the corresponding bits of the following registers, preferably in the same sequence:

For Go Mode:

COMMAND2: Set the type of transfer i.e. Receive (Bit31), transmit (Bit 30)

COMMAND1: Set the bit_length (bits [4:0]), number of words (bits [9:5]), mode (mode0, mode1, mode2 and mode3) as below

Table 99. Go Mode Command 1 for Different Modes

Mode	IDLE_SCLK	CK_SDA	Notes
Mode0	00	0	Clock is positive polarity and the data is latched on the positive edge of SCK
Mode1	00	1	Clock is positive polarity and the data is latched on the negative edge of SCK
Mode2	01	0	Clock is negative polarity and the data is latched on the negative edge of SCK
Mode3	01	1	Clock is negative polarity and the data is latched on the positive edge of SCK

TX_FIFO: Write data into TXFIFO

COMMAND1: Enable the Go Bit (bit 30)

For DMA mode:

COMMAND2: Set the type of transfer i.e. Receive (Bit31), transmit (Bit30)

COMMAND1: Set the bit_length (bits [4:0]), and mode as described above. Please note that the bit length to be set accordingly depending on the PACK_SIZE (bits [22:21] when packed mode(Bit 20)is set in the DMA_CTL register

```
PACK_SIZE       BIT_LENGTH
00              0x03
01              0x07
10              0x0f
11              0x1f
```

DMA_CTL: Set the DMA_BLOCK_LENGTH (bits [15:0]), RX_TRIG (bits [19:18]), TX_TRIG (bits [17:16]). For interrupts set IE_RXC (bit27), IE_TXC(bit26) to 1. For packed mode set PACK_SIZE (bits [22:21]) and PACKED (bit 20).

> **Note:** PACKED should be set after programming all the other parameters (except DMA_EN) in the COMMAND,COMMAND2 and DMA_CTL registers after setting all the parameters set DMA_EN(bit 31) to 1.

The DMA request signals to the APBDMA controller are enabled by the SLINK COMMAND2.TXEN and RXEN bits. Always program the DMA channels first. Enable the COMMAND2.TXEN and RXEN bits last.

35.3.2 SLINK_COMMAND_0

Used for the setting of bit_length, number of words, and for selecting the transfer mode. Chip Select can also be selected to be in HW mode as SW mode with both active high and active low polarities for supporting devices with varying CS polarities.

Serial Link-2 Sub Block Command Register

Offset: 000h | Read/Write: R/W | Reset: 0b0000xx000000xx00xx00000000000000

Bit	Reset	Description
31	0x0	ENB: RD/WD access to Data Register would start the next transfer. (This allows continuous Receive via RD of Buffer and Automated Transmit per WD of Buffer Register) 0 = DISABLE 1 = ENABLE
30	0x0	GO: Program 1 after all the other bits in the COMMAND2 and COMMAND are programmed to start the transfer HW clears this bit automatically after the transfer is done Clearing of the bit by SW will stop the Shifter and latch the partial data into buffer. 0 = STOP 1 = GO
28	0x0	M_S: 1 = Master Mode (internal Clock) 0 = Slave Mode (external Clock) 0 = SLAVE 1 = MASTER
25:24	0x0	IDLE_SCLK: 11 = Pull High 10 = Pull Low 01 = Driven High 00 = Driven Low (default) 0 = DRIVE_LOW 1 = DRIVE_HIGH 2 = PULL_LOW 3 = PULL_HIGH
23	0x0	CS_POLARITY3: 1 = CS3 active high 0 = CS3 active low 0 = LOW 1 = HIGH
22	0x0	CS_POLARITY2: 1 = CS2 active high 0 = CS2 active low 0 = LOW 1 = HIGH
21	0x0	CK_SDA: 1 = Rising Edge 0 = Falling Edge (default) 0 = FALLING_EDGE 1 = RISING_EDGE
20	0x0	CS_POLARITY1: 1 = CS1 active high 0 = CS1 active low 0 = LOW 1 = HIGH
17:16	0x0	IDLE_SDA: 11 = Pull High 10 = Pull Low 01 = Driven High 00 = Driven Low 0 = DRIVE_LOW 1 = DRIVE_HIGH 2 = PULL_LOW 3 = PULL_HIGH
13	0x0	CS_POLARITY: 1 = CS active high 0 = CS active low 0 = LOW 1 = HIGH

Bit	Reset	Description
12	0x0	CS_VALUE: 1 = CS is high 0 = CS is low 0 = LOW 1 = HIGH
11	0x0	CS_SW: 1 = CS controlled by SW 0 = CS controlled by hardware 0 = HARD 1 = SOFT
10	0x0	BOTH_EN: 1 = both lines transmit/receive; 0 = one line transmit and other receive 0 = DISABLE 1 = ENABLE
9:5	0x0	WORD_SIZE: 31 = Thirty Two words (Max)
4:0	0x0	BIT_LENGTH: 31 = Thirty Two bit Transfers (Max)

35.3.3 SLINK_COMMAND2_0

SLINK COMMAND2 register -- This is in general used to select the type of transfer (TX/RX/Both). Select LSBFE for transmitting LSB first.

Serial Link-2 Sub Block Command2 Register

Offset: 004h | Read/Write: R/W | Reset: 0b000000xx0000000xxxx0000000x0xx00

Bit	Reset	Description
31	0x0	RXEN: Receive enable 0 = DISABLE 1 = ENABLE
30	0x0	TXEN: Transmit enable 0 = DISABLE 1 = ENABLE
29	0x0	SPC0: 1 = bi directional mode; 0= Normal Mode 0 = NORMAL 1 = BIDIR
28:26	0x0	WAIT_PACK_INT: number of cycles between two packs in the DMA. Use of this field is deprecated. Use INT_SIZE 8 = number of cycles between 2 packs (Max)
23:22	0x0	FIFO_REFILLS: Number of transfers the CS should stay low for word sizes more than 32. This will enable to do the transfer of word sizes > 32 without using APB-DMA 0x00: For word_sizes 1 to 32 0x01: For word_sizes 33 to 64 0x10: For word sizes 65 to 96 0x11: For word sizes 97 to 128 0 = REFILL0 1 = REFILL1 2 = REFILL2 3 = REFILL3
21:20	0x0	SS_SETUP: number of cycles CS should stay inactive between packets 4 = number of cycles in setup for chip select (Max)

Bit	Reset	Description
19:18	0x0	SS_EN: 11 = chip select3; 10 = chip select2; 01 = chip select1; 00 = chip select0 (default) 0 = CS0 1 = CS1 2 = CS2 3 = CS3
17	0x0	CS_ACTIVE_BETWEEN: 1 = CS active between two packets. 0 = CS inactive between two packets 0 = LOW 1 = HIGH
12:8	0x0	INT_SIZE: number of IDLE cycles between two packets 31 = thirty two cycles between 2 packets
7	0x0	MODFEN: 1 = Enable Modef; 0 = Disable Modef (default) 0 = DISABLE 1 = ENABLE
6	0x0	BIDIROE: When set to 1 SPI uses only one data line (mosi/miso) for Tx and Rx depending on Master/Slave mode. This has effect only when SPC0 is set to 1 1 = ENABLE Enable Output buffer 0 = DISABLE Disable Output buffer (default)
4	0x0	SPIE: 1 = ENABLE Enable SPIE interrupt 0 = DISABLE Disable SPIE interrupt
1	0x0	SSOE: 1 = Enable; 0 = Disable (default) 0 = DISABLE 1 = ENABLE
0	0x0	LSBFE: 1 = FIRST Transmit/Receive LSB first 0 = LAST Transmit/Receive LSB last

35.3.4 SLINK_STATUS_0

SLINK STATUS register: Read this register to know the status of the transfer. Error bit is set whenever errors such as Underflow/Overflow are encountered.

Serial Link-2 Sub Block Status/Control Register

Offset: 008h | Read/Write: R/W | Reset: 0b00000000101000x00000000000000000

Bit	Reset	Description
31	0x0	BSY: 1 = BUSY: Controller is Busy 0 = IDLE: Controller is Free
30	0x0	RDY: 1 = READY: Controller is Ready for transfer 0 = NOT_READY: Controller is Busy. Write 1 to clear the flag
29	0x0	ERR: Will be set to 1 by HW when Errors such as Underflow/overflow occur. Write 1 to clear the flag 0 = OK 1 = ERROR

Bit	Reset	Description
28	0x0	SCLK: SCLK input signal State 0 = LOW 1 = HIGH
27	0x0	RX_FLUSH: Flush the RX FIFO 0 = NOP 1 = FLUSH
26	0x0	TX_FLUSH: Flush the TX FIFO 0 = NOP 1 = FLUSH
25	0x0	RX_OVF: RX FIFO Overflow 0 = OK 1 = ERROR
24	0x0	TX_UNF: TX FIFO Underflow 0 = OK 1 = ERROR
23	0x1	RX_EMPTY: RX FIFO Empty 0 = NOT_EMPTY 1 = EMPTY
22	0x0	RX_FULL: RX FIFO Full 0 = NOT_FULL 1 = FULL
21	0x1	TX_EMPTY: TX FIFO Empty 0 = NOT_EMPTY 1 = EMPTY
20	0x0	TX_FULL: TX FIFO Full 0 = NOT_FULL 1 = FULL
19	0x0	TX_OVF: TX FIFO Overflow 0 = OK 1 = ERROR
18	0x0	RX_UNF: RX FIFO Underflow 0 = OK 1 = ERROR
16	0x0	MODF: Mode fault 0 = OK 1 = ERROR
9:5	0x0	WORD: In GO mode indicates number of words transferred (word count)
15:0	0x0	BLK_CNT: Number of blocks transferred (BLOCK count) during DMA
4:0	0x0	COUNT: In Go mode indicates number of bits transferred (bit count)

35.3.5 SLINK_MAS_DATA_0

Serial Link-2 Sub Block Transmit Data Page Buffer Register

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	MASTER_BUFFER: Tx/Rx Shift Pattern

35.3.6 SLINK_SLAVE_DATA_0

Serial Link-2 Sub Block Slave Data Page Buffer Register

Offset: 014h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	SLAVE_BUFFER: Tx/Rx Shift Pattern

35.3.7 SLINK_DMA_CTL_0

SLINK DMA Control Register: Used in the DMA transfers. Set packed mode transfers and the trigger levels in this register.

Serial Link-2 Sub Block DMA Control Register

Offset: 018h | Read/Write: R/W | Reset: 0b00xx00xxx00000000000000000000000

Bit	Reset	Description
31	0x0	DMA_EN: 1 = ENABLE DMA mode is enabled, 0 = DISABLE DMA disabled
30	0x0	CONT: 1 = continuous mode is enabled, 0 = continuous mode disabled 0 = DISABLE 1 = ENABLE
27	0x0	IE_RXC: Interrupt enable on receive completion. 1 = ENABLE Enable interrupt generation at the end of a receive transfer. 0 = DISABLE Disable interrupt generation for receive.
26	0x0	IE_TXC: Interrupt enable on transmit completion. 1 = ENABLE Enable interrupt generation at the end of a transmit transfer. 0 = DISABLE Disable interrupt generation for transmit.
22:21	0x0	PACK_SIZE: Specifies the packet size during the DMA mode 00 = 4 bits in a pack 01 = 8 bits in a pack 10 = 16 bits in a pack 10 = 32 bits in a pack 0 = PACK4 1 = PACK8 2 = PACK16 3 = PACK32
20	0x0	PACKED: Packed mode enable bit. 1 = Packed mode is enabled. This is only valid if BIT_LENGTH in SBCX_COMMAND register is set to 3, 7, 15 or 31. When enabled, all 32-bits of data in the FIFO contain valid data packets of either 8-bit or 16-bit length. 0 = Packed mode is disabled. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
19:18	0x0	RX_TRIG: Receive FIFO trigger level. 00 = 1 word. DMA trigger is asserted whenever there is at least 1 word in the RX FIFO. 01 = 4 words. DMA trigger is asserted when there are at least 4 words in the RX FIFO. 10 = 8 words. DMA trigger is asserted when there are at least 8 words in the RX FIFO. 11 = 16 words. DMA trigger is asserted when there are at least 16 words in the RX FIFO. 0 = TRIG1 1 = TRIG4 2 = TRIG8 3 = TRIG16
17:16	0x0	TX_TRIG: Transmit FIFO trigger level. 00 = 1 word. DMA trigger is asserted whenever there is at least 1 word in the TX FIFO. 01 = 4 words. DMA trigger is asserted when there are at least 4 words in the TX FIFO. 10 = 8 words. DMA trigger is asserted when there are at least 8 words in the TX FIFO. 11 = 16 words. DMA trigger is asserted when there are at least 16 words in the TX FIFO. 0 = TRIG1 1 = TRIG4 2 = TRIG8 3 = TRIG16
15:0	0x0	DMA_BLOCK_SIZE: N = N+1 packets number of packets should be aligned in the packed mode transfers. packed mode --> Number of packets 3 multiple of 8 7 multiple of 4 15 multiple of 2 31 from 0 to N.

35.3.8 SLINK_STATUS2_0

Serial Link-2 Status2 Register

SLINK Status2 Register: Contains the empty count for the transmit FIFO and the full count for receive FIFO

Offset: 01ch | Read/Write: R/W | Reset: 0b0000000xxxxxxxxxxx00000000xxxxxx

Bit	R/W	Reset	Description
31:26	RW	0x0	TAP_DELAY: Indicates the tap delay for the SPI external clock
25	RW	0x0	SBC_CE: Indicates the SBC clock enable to CAR
24	RO	X	CS_INACTIVE: 0 = CS active during continuous, 1 = CS inactive during continuous
23	RO	X	FRAME_END: 1 = End of frame
21:16	RO	X	RX_FIFO_FULL_COUNT: Indicates the number of words in the receive FIFO
13:10	RW	0x0	BETWEEN_CS: Indicates the number of cycles between two transfers
9:6	RW	0x0	HOLD_TIME: Indicates the number of cycles in hold time for CS staying asserted at the end of transfer
5:0	RO	X	TX_FIFO_EMPTY_COUNT: Indicates the number of empty slots in the transmit FIFO

35.3.9 SLINK_TX_FIFO_0

Serial Link-2 Sub Block TX FIFO Buffer Register

Offset: 100h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000 | Default: 0000.0000

Bit	Reset	Description
31:0	0x0	TX_FIFO_REGISTER: Tx/Rx Shift Pattern

35.3.10 SLINK_RX_FIFO_0

Serial Link-2 Sub Block RX FIFO Buffer Register

Offset: 180h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000 | Default: 0000.0000

Bit	Reset	Description
31:0	0x0	RX_FIFO_REGISTER: Tx/Rx Shift Pattern

36.0 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 uses two wires for this--one for ground, and the other for power and data.

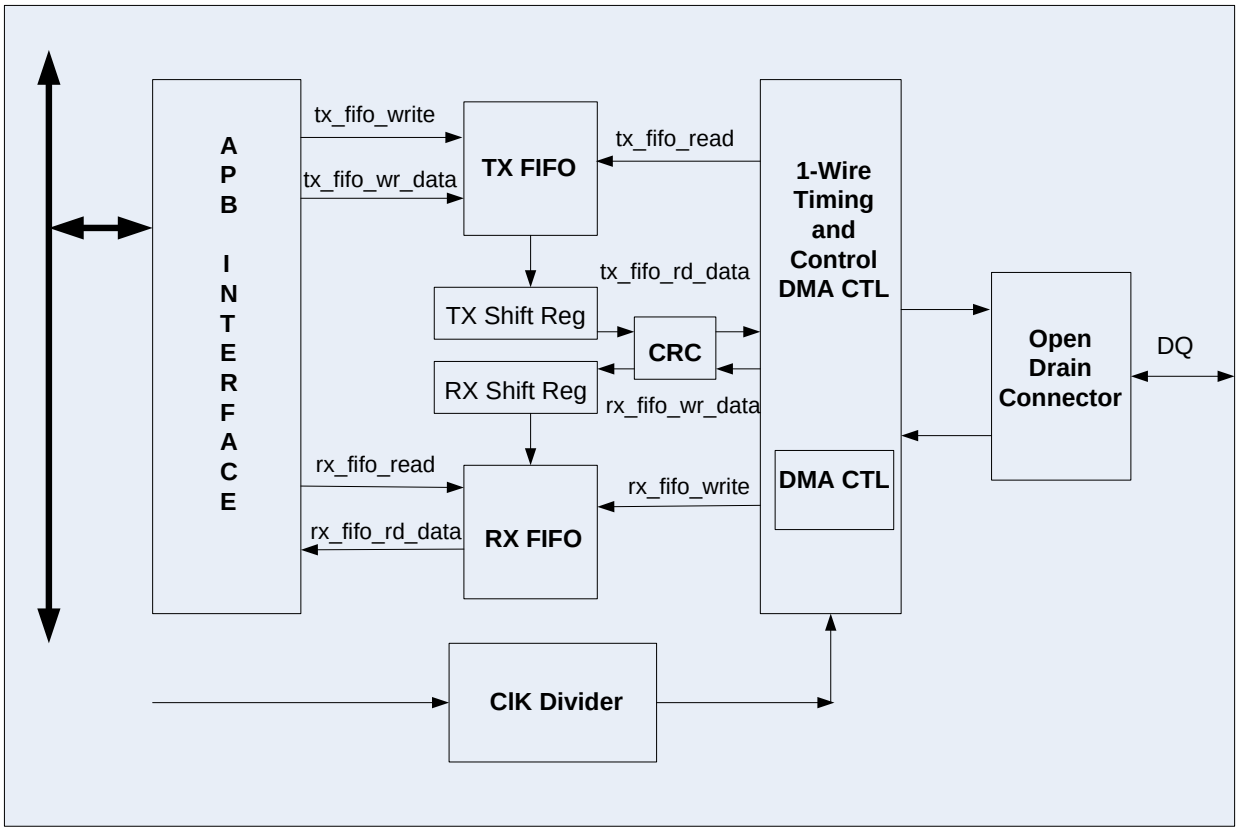
In Tegra® 3 devices, the one wire protocol is intended to communicate with battery controller chips.

Features

- Independent RX and TX FIFO's
- FIFO depth of 32 x 32 bits
- Hardwired implementation of one wire protocol to eliminate need for external bridge chip
- Software programmable registers
- Software readable status registers
- Receive transmission with interrupt-based operation
- 1 MHz device clock required. Capable of running at higher frequencies provided timing parameters are programmed
- Supports deglitch
- Supports Byte transfers or 1 Bit transfers
- Supports following commands: Read ROM, Skip ROM, Read Mem, Read Status, Read Data/Generate 8 bit CRC, Write Memory, Write Status
- Supports CRC 8/16 bit implementation
- Supports different battery devices, up to a memory size of 256K in byte transfer
- OWR generic controller support added in Tegra 3 with device DS2784. The features supported in DS2784 are:
  - Command
  - Command and Address
  - Command, Address and Write Command
  - Command, Address and Read Command

36.1 Functionality

Figure 100. OWR Block Diagram



Specific command sequences as per the OWR protocol have to be followed for any data transaction over 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.

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)

The other commands can be used with 1-bit read/write. Contact your NVIDIA FAE for more details.

36.1.1 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 Control Register).When this bit is enabled after the data is read from the EPROM, the device sends CRC to the controller .The controller then checks the received CRC against the calculated CRC and sets an crc_error flag in case of mismatch. The CRC can be 8 bit /16 bit depending on 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 & continues with the command operation.

For other commands, i.e., 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 16bit is also supported in some battery devices.

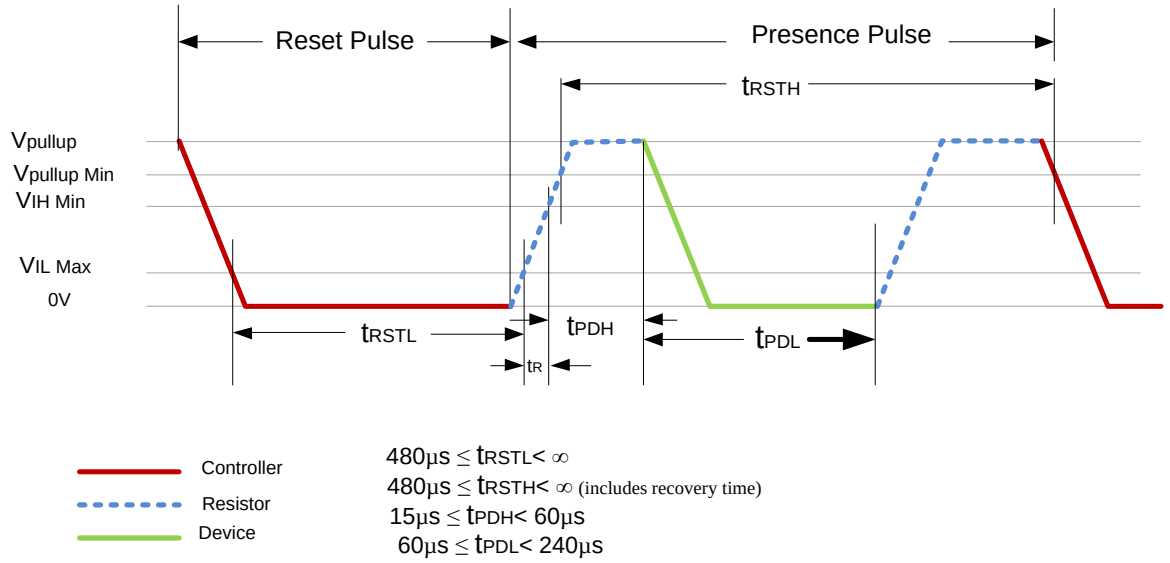
36.1.2 OWR Data Transfer Sequence

The OWR protocol consists of five types of data transfer:

- Reset & 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 the Figure 101 below.

Figure 101. Initialization Sequence



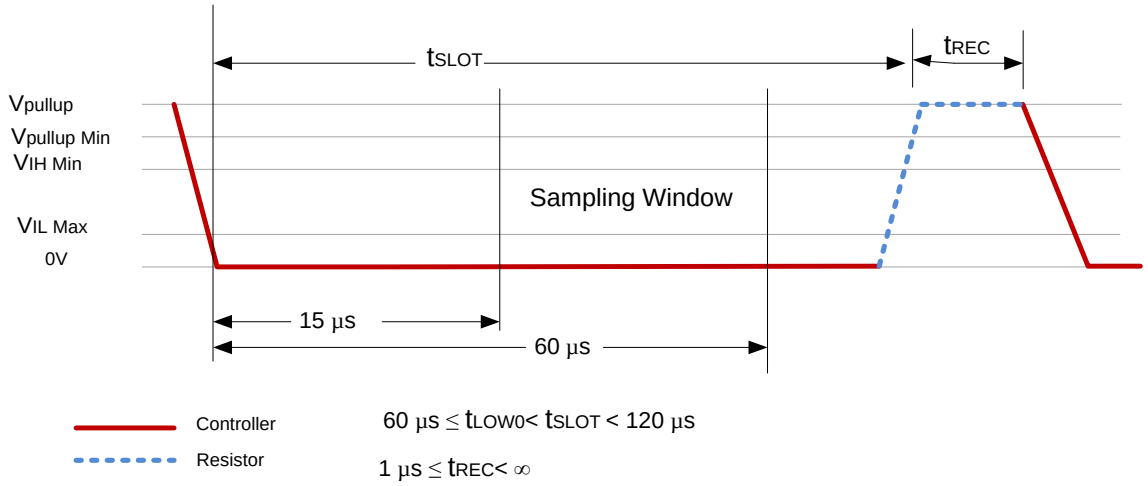
The reset pulse begins the initialization sequence. It is initiated when the GO bit in the control register (Control[GO]) is set.

36.1.2.1 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 scratch pad any time between t_{LOW0} and t_{SLOT} duration as shown in Figure 102 below

Figure 102. 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 103). Similarly during a read slot the controller can read the data from the device in sampling window (Figure 104)

Figure 103. Write 1 Timing

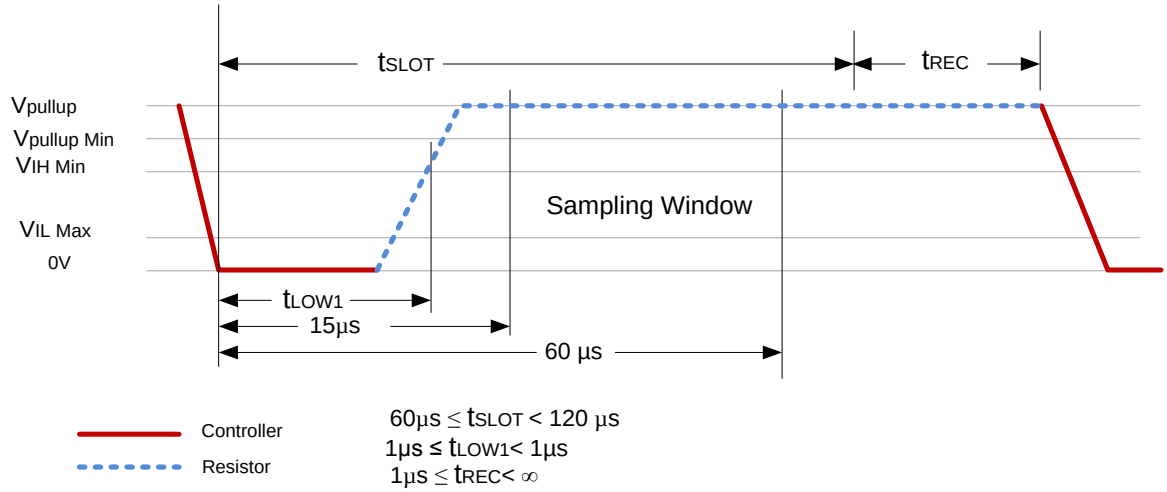
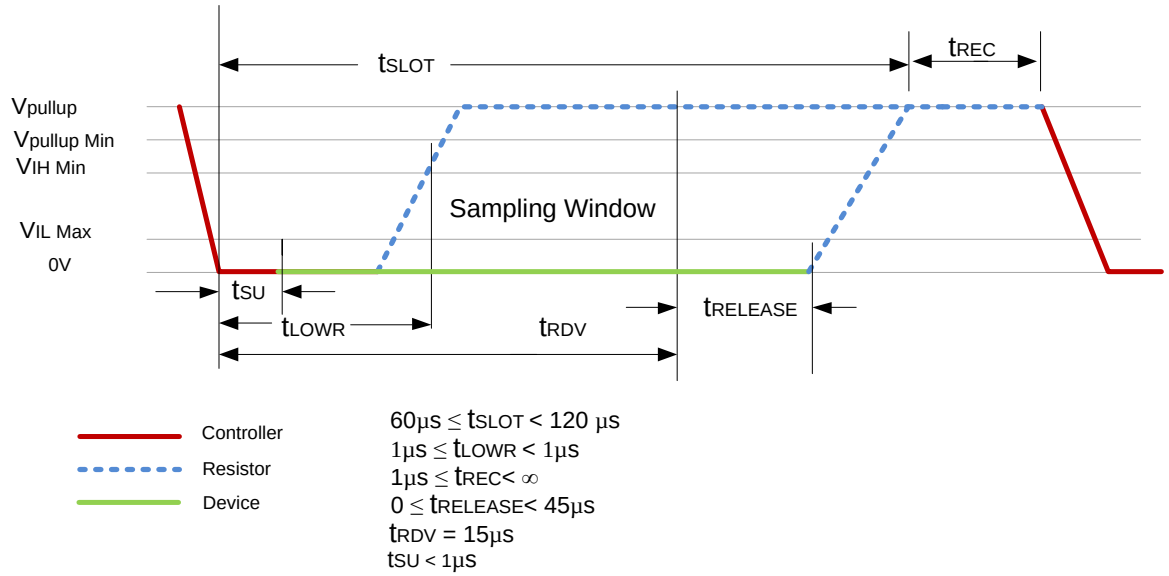


Figure 104. Read Timing

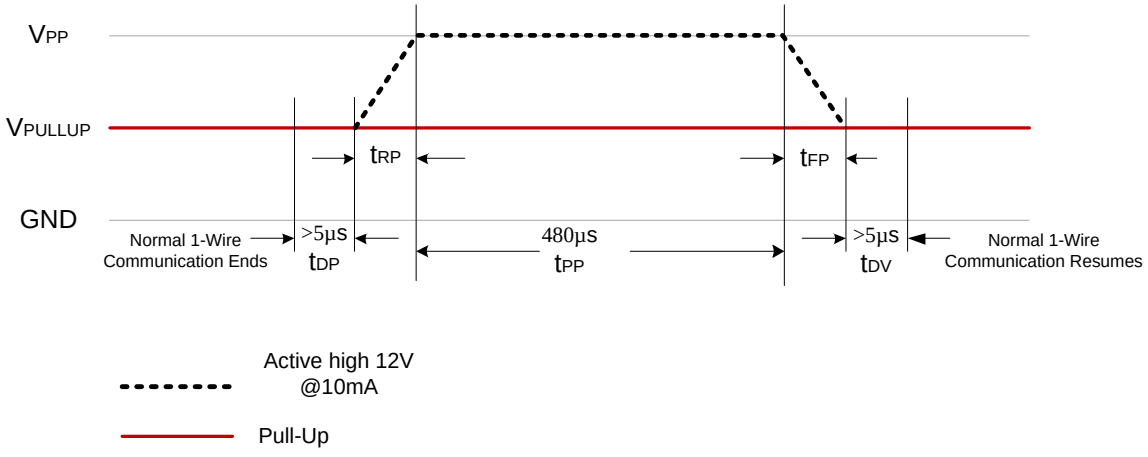


36.1.2.2 Programming Pulse

During a write memory command, data is first copied to the scratch pad. To shift the data from the scratch pad 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 transfer the controller controls the generation of programming pulse.
- In single bit Write mode programming pulse is generated using GPIO's
- In Programming pulse window OWR bus is controlled by 12V voltage providing minimum of 10mA of current.

Figure 105. Programming Timing Pulse

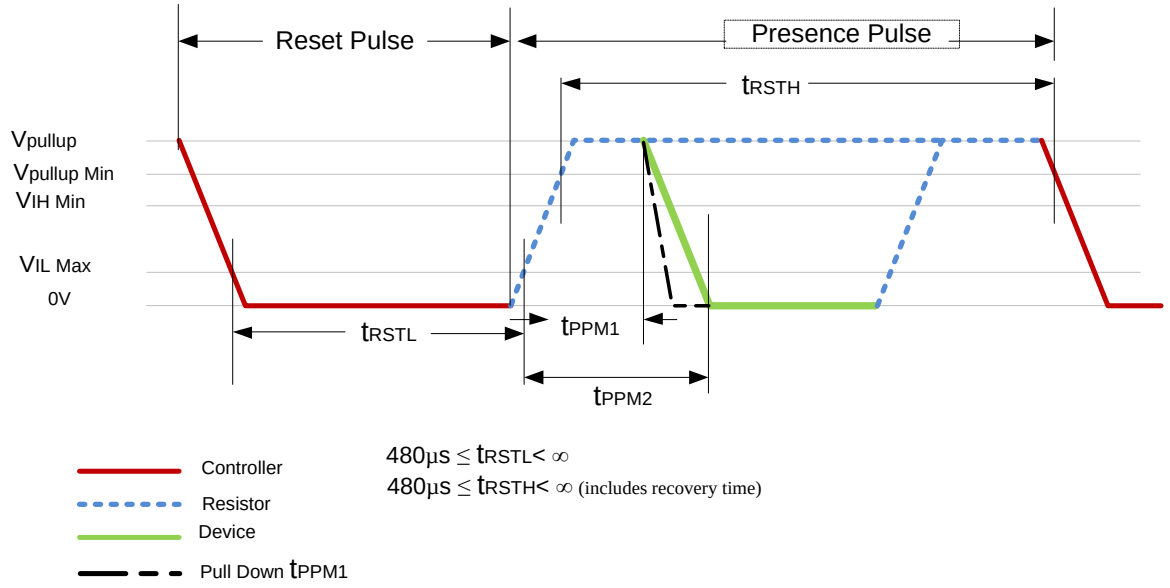


36.1.2.3 Presence-Pulse Masking (PPM)

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} till t_{PPM2}. At t_{PPM2} battery device further pulls down to low.

Figure 106. Presence-Pulse Masking



36.1.3 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 has been done without any interruptions.

Error signals indicate an error has occurred and data transfer should start again. Error can be CRC err, presence error, memory write error etc.

Overflow/underrun indicates the FIFO status.

Data request indicates FIFO's are ready to accept or send the data.

All the Interrupts can be controlled by Interrupt MASK enables and Interrupt SET register.

36.1.4 Error handling

If OWR controller receives an error like 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 doesn't stops on CRC error

CTL.BYPASS_DGLITCH: Stops the execution of deglitch logic

CTL.PRESENCE_PULSE_MASKING: Transfer doesn't stops on presence error.

36.1.5 Clock Control

OWR controller clocking has following clock source options:

Option 0: pllP_out0 = 432MHZ (Fixed 432 MHz directly from PLLP)

Option 1: pllC_out0 = Up to 600MHz, Programmable clock output directly from PLLC.

Option 2: pllM_out0 = Up to 400MHz, 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.

36.1.6 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 reg presence_sample_clk for reset initialization and read_data_sample_clk for read time slots.

There is a limitation to the value programmed in Presence/read_data sample clk: the hardware requires 6 clock cycles to implement deglitch generation logic (Synchronizer output requires 2 clks and deglitch 1 clk. If there is a glitch on dq line, HW samples the data on next clk edge, 2 clks later, to store the sampled data 1 clk pulse).

Presence sample clk = Tpdl - 6 clks

Rd_data_sample_clk = Trdv - 6 clks

36.1.7 Read/Write 1 Bit

The controller also supports sending or receiving data bit by bit. In 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
- check CRC and
- preserves the received data

The hardware does not preserve any data in registers, except the received 1 bit data, in RD_SAMPLE_DATA register.

The controller can only send the Time slots (Wr1/Wr0/Rd) and store the received sampled data in a reg. 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.

36.1.7.1 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 Memory Command.

Programming Rom Cmd

Transmit LSB first to send ReadRom Command (33h, 0011-0011).

Time Slots

- LSB of ReadRom command is 1'b1. Program CTRL.WR1_BIT in control register. After controller generates write-1 time slot, wait for the write-1 time slot to be completed by looking into INTR_STATUS.BIT_TRANSFER_DONE.
- After this is complete, program the next bit (in this case it's again 1'b1). Program CTRL.WR1_BIT in control register. After Controller generates write-1 time slot, wait for the write-1 time slot to be completed by looking into INTR_STATUS.BIT_TRANSFER_DONE
- Program the next bit then (in this case it is 1'b0). Program CTRL.WR0_BIT in control register. After controller generates write-0 time slot, wait for the write-0 time slot to be completed by looking into INTR_STATUS.BIT_TRANSFER_DONE

Repeat this for the remaining bits.

- After the ReadRom Command is sent, read 64 bits rom data. To receive 64 bits, 64 read time slots have to be sent. To generate Read time slot, program CTRL.RD_BIT in control register. After the controller generates read time slot, the read bit is stored in READ_SAMPLED_BIT. Sampled Bit should always be read after the bit transfer is done.
- Next, program the CTRL.RD_BIT, to read 2nd bit. Read time slot is generated by the controller. Wait for done and read the sampled bit.

Repeat the sequence for remaining bits.

After receiving all 64 bits, Memory data has to be read.

Reading EPROM data using ReadMem Command

- To read Memory data, first send memory cmd followed by memory address. After memory cmd and address is sent, verify the data sent. This can be done by checking CRC of cmd and address sent. To receive CRC, send 8 read-slots, read the CRC data and check with computed value if it matches. 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 endoffset. After reading EPROM data, read CRC and check the received data if CRC is required. Normally CRC is embedded within the EPROM data. To receive 8 bit CRC data send 8 read time slots.

36.2 Programming Guidelines

36.2.1 Programming Registers

All the registers should be programmed and CTL.GO bit should be programmed to start the controller.

Registers to Program Prior to the Control Register

- **All the timing registers**: Program the timing values according to battery device data sheet
- **Command reg**: Program the device Rom Cmd, Mem Cmd, 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 Reg**: Programming this register depends on the type of execution required.

Program the G0 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.

36.2.2 Byte Data Transfer Sequence

Reset Initialization → Rom Cmd → Mem Cmd → Reset Initialization → Rom Cmd → Mem Cmd etc

In case of an error, the entire sequence should start again from the beginning:

Reset Initialization → Error → Reset Initialization →Rom Cmd → Error → Reset Initialization → Rom Cmd → Mem Cmd etc.

36.3 OWR Registers

36.3.1 OWR_CONTROL_0

This register is the main control register. It should be configured last after configuring all other settings.

OWR control Register

Offset: 000h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31	0x0	RD_BIT: if 0 no transfer is done. If 1 read time slot is executed. This bit is a write only. Read to this register 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. This bit is a write only. Read to this register 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. This bit is a write only. Read to this register will return 0 0 = NO_TRANSFER 1 = TRANSFER_ONE
28	0x0	BY_PASS_CRC_ERR: this bit is set to 1, if transfer needs to continue on crc err else on err transfer stops, and again transfer starts on setting rpp reset(go bit) 0 = STOP_TRANSFER_ON_CRC_ERR 1 = CONTINUE_TRANSFER_ON_CRC_ERR
27	0x0	BY_PASS_DGLITCH: This bit is used to bypass the deglitch logic, If 1, just takes the sync output If 0, looks for any glitch in the sample window for at least 1us, Deglitch requires a minimum of 6 clks(2 for sync, 2 for deglitch, if glitch, checks for 2 more clks, still glitch exists, err interrupt is asserted and data transfer should start from first) 0 = START_DGLITCH 1 = NO_DGLITCH
26:23	0x0	RD_DATA_SAMPLE_CLK: read data sample window, master samples the data_in which should be less than or equal to (tlow1 - 6) clks 6 clks are used for Deglitch, if Deglitch bypassed 3 clks should be enough
22:15	0x0	PRESENCE_SAMPLE_CLK: presence pulse sample clk, master samples the data_in which should be less than or equal to (tpdl - 6) clks 6 clks are used for dglitch, if Deglitch bypassed 3 clks should be enough
14	0x0	RD_MEM_CRC_REQ: This bit is set to 1, if CRC is required for read memory cmd at end of memory 0 = NO_CRC_READ

Bit	Reset	Description
		1 = CRC_READ
13:9	0x0	RX_FIFO_ATTEN_LEVEL: Receive FIFO attention level 000 = 1 word, fifo req is asserted when least one word full in the fifo 001 = 2 word, fifo req is asserted when least 2 words full in the fifo etc.
8:4	0x0	TX_FIFO_ATTEN_LEVEL: Transmit fifo attention level 000 = 1 word, fifo req is asserted when least one word empty in the fifo 001 = 2 word, fifo req is asserted when least 2 words empty in the fifo etc.
3	0x0	CRC_16BIT_EN: if set to 1 16bit CRC is executed if set to 0 8bit 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 through byte 0 = BYTE_TRANSFER_MODE 1 = BIT_TRANSFER_MODE
1	0x0	PRESENCE_PULSE_MASKING: when set, dq is driven to low by master before the slave does clearing this bit disables the ppm 0 = NO_PPM 1 = START_PPM
0	0x0	GO: Generate Reset Presence Pulse This bit is a write only read to this register will return 0this bit should be programed after all the registers are programed 0 = NO_PRESENCE_PULSE 1 = START_PRESENCE_PULSE

36.3.2 OWR_COMMAND_0

Rom Cmd, Mem Cmd and Mem Addr should be program at the same time.

OWR Command Register

Offset: 004h | Read/Write: R/W | Reset: 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: one wire MEM commands
7:0	0x0	ROM_CMD: one wire ROM commands

36.3.3 OWR_EPROM_0

Indicates the number of bytes to transfer, supports up to 256K EPROM size. MEM_ADDR and MEMORY_BYTES_TRANSFER/STATUS_BYTES_TRANSFER should always be in sync as in if EPROM end offset address is 7f, and we want to read data from 4th location address of EPROM.

Programming should be in the following way:

MEM_ADDR = 0x4;

MEMORY_BYTES_TRANSFER = 0x7B; (0x4 - 0x7f) similarly for status bytes.

OWR Eprom Size offset

Offset: 008h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	STATUS_BYTES_TRANSFER: Number of EPROM Status bytes to transfer Mem_Addr - Status bytes end address

Bit	Reset	Description
15:0	0x0	MEMORY_BYTES_TRANSFER: Number of EPROM memory bytes to transfer, Mem_Addr - EPROM end address

36.3.4 OWR_WR_RD_TCTL_0

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.

OWR Write Read Timing Control Register

Offset: 00ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
29:28	0x0	TSU: Read Data Setup, Tsu = N owr clks, Range = tsu < 1
27:22	0x0	TRELEASE: Release one wire Time, Trelease = N owr clks, Range = 0 <= trelease < 45
21:18	0x0	TRDV: Read data valid time, Trdv = N+1 owr clks, Range = Exactly 15
17:11	0x0	TLOW0: Write Zero time Low, Tlow0 = N+1 owr clks, Range = 60 <= tlow0 < tslot < 120
10:7	0x0	TLOW1: Write one time Low, or TLOWR both are same Tlow1 = N+1 owr clks, Range = 1 <= tlow1 < 15 TlowR = N+1 owr clks, Range = 1 <= tlowR < 15
6:0	0x0	TSLOT: Active time slot for write or read data, Tslot = N+1 owr clks, Range = 60 <= tslot < 120

36.3.5 OWR_RST_PRESENCE_TCTL_0

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

OWR Reset Presence Timing Control Register

Offset: 010h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	TPDL: PRESENCE_DETECT_LOW Tpdl = N owr clks, Range = 60 <= tpdl < 240
23:18	0x0	TPDH: PRESENCE_DETECT_HIGH Tpdh = N+1 owr clks, Range = 15 <= tpdh < 60
17:9	0x0	TRSTL: RESET_TIME_LOW Trstl = N+1 owr clks, Range = 480 <= trstl < infinity
8:0	0x0	TRSTH: RESET_TIME_HIGH, Trsth = N+1 owr clks, Range = 480 <= trsth < infinity

36.3.6 OWR_PPM_CORRECTION_TCTL_0

Drives the DQ line to low before slave does

OWR Presence Pulse Masking Timing Control

Offset: 014h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:6	0x0	TPPM2: PRESENCE PULSE MASK STOP
5:0	0x0	TPPM1: PRESENCE PULSE MASK START

36.3.7 OWR_PROG_PULSE_TCTL_0

Controls the 12V programming pulse used to write data to EPROM. Timings are in microseconds.

OWR Program Pulse Timing Control Register

Offset: 018h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	TPP: Program Pulse Width Tpp = N OWR clks Range = 480 to 5000
15:12	0x0	TFP: Program Voltage Fall Time Tfp = N OWR clks Range = 0.5 to 5
11:8	0x0	TRP: Program Voltage Rise Time Trp = N OWR clks Range = 0.5 to 5
7:4	0x0	TDV: Delay to verify Tdv = N OWR clks, Range = > 5
3:0	0x0	TPD: Delay to program Tpd = N+1 OWR clks, Range = > 5

36.3.8 OWR_READ_ROM0_0

Read ROM Cmd Reads 8-bit family code, 48-bit serial number and 8-bit CRC. Total of 64 bits, Register can have a maximum of 32 bits. Read Rom Cmd data is split into 2 registers 1. READ_ROM0 2. READ_ROM1

OWR Read ROM DATA0 Register

Offset: 01ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:8	X	SERIAL_NUM0: Reads the first 24 bits of ROM serial number
7:0	X	FAMILY_CODE: Reads the 8 bit family code of ROM

36.3.9 OWR_READ_ROM1_0

OWR Read ROM DATA1 Register

Offset: 020h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:24	X	CRC_BYTE: Reads the 8-bit CRC code of ROM
23:0	X	SERIAL_NUM1: Reads the next 24 bits of ROM serial number

36.3.10 OWR_INTR_MASK_0

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.

OWR INTR Mask Register

Offset: 024h | Read/Write: R/W | Reset: 0b00000000000000

Bit	Reset	Description
13	0x0	BIT_TRANSFER_DONE_INT_EN: 0 = DISABLE 1 = ENABLE
12	0x0	RXFIFO_DATA_REQ_INT_EN: 0 = DISABLE

Bit	Reset	Description
		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
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

36.3.11 OWR_INTR_STATUS_0

OWR Status Register

Offset: 028h | Read/Write: R/W | Reset: 0b0xx00000000000

Bit	R/W	Reset	Description
13	RW	0x0	BIT_TRANSFER_DONE: This bit is set when transfer of each bit done this is set on in one bit transfer mode software writes 1 to clear this bit 0 = NOT_DONE 1 = DONE
12	RO	X	RXFIFO_DATA_REQ: RX FIFO data req 0 = RX_NOT_RDY 1 = RX_RDY
11	RO	X	TXFIFO_DATA_REQ: TX FIFO data req 0 = TX_NOT_RDY 1 = TX_RDY

Bit	R/W	Reset	Description
10	RW	0x0	DGLITCH: This bit is set when data is not stable for at least 1us, Software writes a 1 to clear this bit. if deglitch detected data transfer should start from 1st. 0 = DGLITCH_NOT_DETECTED 1 = DGLITCH_DETECTED
9	RW	0x0	RXF_UNR: RX FIFO Under run: RO. 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: RO. 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 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 master has received the rom data from battery. Software writes a 1 to clear it. 0 = NOT_DONE 1 = DONE
5	RW	0x0	PRESENCE_DONE: This indicates the presence done, 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
3	RW	0x0	ERR_CMD: ERROR CMD: Indicates error command written in the register. Software writes a 1 to clear it. 0 = CORRECT_CMD 1 = ERROR_CMD
2	RW	0x0	MEM_WR_ERR: MEM WR ERROR: Indicates 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 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 not found 0 = SLAVE_DETECTED 1 = NO_SLAVE_DETECTED

36.3.12 OWR_INTR_SOURCE_0

This is a read-only register which returns the AND of Interrupt Source and Interrupt Mask registers.

OWR INTR Source Register

Offset: 02ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxx

Bit	Reset	Description
13	X	BIT_TRANSFER_DONE: 0 = NOT_DONE

Bit	Reset	Description
		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
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

36.3.13 OWR_INTR_SET_0

This is a write-only register which can be used to set the interrupt status bit. A write to this register causes the bits in status register to be set 1 Read always returns 0.

Interrupt Set Register

Offset: 030h | Read/Write: R/W | Reset: 0b0xx00000000000

Bit	Reset	Description
13	0x0	BIT_TRANSFER_DONE: 0 = NOT_DONE 1 = DONE

Bit	Reset	Description
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
1	0x0	CRC_ERR: 0 = N0_ERROR 1 = ERROR
0	0x0	PRESENCE_ERR: 0 = SLAVE_DETECTED 1 = NO_SLAVE_DETECTED

36.3.14 OWR_STATUS_0

OWR Status Register

Offset: 034h | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxxxxx00xxxxx

Bit	R/W	Reset	Description
23	RO	X	READ_SAMPLED_BIT: the bit is valid only RD_BUSY is cleared 0 = READ_ZERO 1 = READ_ONE
22	RO	X	RD_BUSY: READ : This is a self-clearing bit and is cleared when read time slot completes on read sequence the sampled read bit is stored in READ_BIT 0 = IDLE 1 = BUSY
21	RO	X	WR1_BUSY: WRITE1 : This is a self-clearing bit and is cleared when write one time slot completes on write sequence 1 is transferred 0 = IDLE 1 = BUSY
20	RO	X	WR0_BUSY: WRITE 0 : This bit is self-clearing, and is cleared when write zero time slot completes on write sequence 0 is transferred 0 = IDLE

Bit	R/W	Reset	Description
			1 = BUSY
19	RO	X	RPP: this is sent reset is set(go), auto cleared on completion of 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: flush the rx fifo, cleared after fifo is empty 0 = DISABLE 1 = ENABLE
5	RW	0x0	TX_FLUSH: flush the tx FIFO, cleared after fifo is empty 0 = DISABLE 1 = ENABLE
4	RO	X	RXF_EMPTY: RX FIFO empty status: RO. Hardware sets this bit to 1 if 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: RO. Hardware sets this bit to 1 if 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: RO. Hardware sets this bit to 1 if 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: RO.Hardware sets this bit to 1 if TX FIFO is full. Otherwise, this bit is set to 0. 0 = NOT_FULL 1 = FULL
0	RO	X	RDY: Ready bit. This bit is set at the end of every transfer and its cleared by hardware when next transfer starts 0 = NOT_READY 1 = READY

36.3.15 OWR_CRC_0

OWR CRC Register

Offset: 038h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	CRC_CALC: CRC calculated by OWR current wr/rd operation
15:0	X	CRC_RECEV: CRC Received on Read Data

36.3.16 OWR_BYTE_CNT_0

OWR BYTE_CNT Register

Offset: 03ch | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	TRANSMITTED: Number of bytes transmitted on wr cmds or address sent
15:0	X	RECEIVED: Number of bytes received on Read Data includes CRC byte count

36.3.17 OWR_TX_FIFO_0

OWR TX FIFO Register

Offset: 040h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	WR_DATA: TX FIFO

36.3.18 OWR_RX_FIFO_0

OWR RX FIFO Register

Offset: 044h | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:0	X	RD_DATA: RX FIFO

36.3.19 OWR_RECOVERY_TIME_TCTL_0

OWR RECOVERY TIME Register

Offset: 04ch | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	TREC: Recovery time slot for write or read data, Trec= N+1 OWR clks

36.3.20 OWR_RST_TIME_TCTL_0

OWR RST PRESENCE HIGH AND LOW TIME Register

Offset: 050h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	TRSTL: RESET_TIME_LOW Trstl = N+1 OWR clks
15:0	0x0	TRSTH: RESET_TIME_HIGH, Trsth = N+1 OWR clks

36.3.21 OWR_RST_PRESENCE_DETECT_TCTL_0

OWR RST PRESENCE DETECT HIGH AND LOW TIME Register

Offset: 054h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	TPDL: PRESENCE_DETECT_LOW Tpdl = N owr clks
15:0	0x0	TPDH: PRESENCE_DETECT_HIGH Tpdh = N+1 owr clks

36.3.22 OWR_WRITE_ONE_ZERO_TCTL_0

OWR WRITE 1 OR WRITE 0 TIME Register

Offset: 058h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	TLOW0: Write Zero time Low Tlow0 = N+1 OWR clks
15:0	0x0	TLOW1: Write one time Low, or TLOWR both are same Tlow1 = N+1 OWR clks TlowR = N+1 OWR clks

36.3.23 OWR_TIME_SLOT_RELEASE_TCTL_0

OWR TIME SLOT OR RELEASE TIME Register

Offset: 05ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	TRELEASE: Release 1-wire Time Trelease = N OWR clks
15:0	0x0	TSLOT: Active time slot for write or read data, Tslot = N+1 OWR clks

36.3.24 OWR_READ_DATA_TCTL_0

OWR WRITE 1 OR WRITE 0 LOW TIME Register

Offset: 060h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	TRDV: Read data valid time Trdv = N+1 OWR clks
15:0	0x0	TSU: Read Data Setup Tsu = N OWR clks

36.3.25 OWR_PROGRAM_PULSE_TCTL_0

OWR PROGRAM PULSE TIMING CONTROL Register

Offset: 064h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:24	0x0	TFP: Program Voltage Fall Time Tfp = N OWR clks
23:16	0x0	TRP: Program Voltage Rise Time Trp = N OWR clks
15:8	0x0	TDV: Delay to verify Tdv = N OWR clks
7:0	0x0	TPD: Delay to program Tpd = N+1 OWR clks

36.3.26 OWR_PROGRAM_PULSE_WIDTH_TCTL_0

OWR PROGRAM PULSE TIMING CONTROL Register

Offset: 068h | Read/Write: R/W | Reset: 0b0000000000000000

Bit	Reset	Description
15:0	0x0	TPP: Program Pulse Width Tpp = N OWR clocks. Range = 480 to 5000

36.3.27 OWR_PPM_TCTL_0

OWR PRESENCE PULSE MASKING TIMING CONTROL

Offset: 06ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:16	0x0	TPPM2: PRESENCE PULSE MASK STOP
15:0	0x0	TPPM1: PRESENCE PULSE MASK START

36.3.28 OWR_SAMPLE_DATA_0

OWR SAMPLE DATA Register

Offset: 070h | Read/Write: R/W | Reset: 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, master samples the data_in which should be less than or equal to (tlow1 - 6) clks. 6 clocks are used for deglitch, if Deglitch 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 clk, master samples the data_in which should be less than or equal to (tpdl - 6) clocks. 6 clks are used for deglitch, if Deglitch bypassed 3 clocks should be enough

36.3.29 OWR_GENERIC_CTL_0

GENERIC OWR COMMAND TRANSFER CONTROL

Offset: 074h | Read/Write: R/W | Reset: 0b00000000000

Bit	Reset	Description
10	0x0	RD_DATA_CRC: 0 = READ 1 = NO_READ
9	0x0	CMD_ADDR_CRC: 0 = READ 1 = NO_READ
8	0x0	DATA_XFER_TWAIT: 0 = WAIT 1 = NO_WAIT
7	0x0	WR_DATA_XFER: 0 = WRITE 1 = NO_WRITE
6	0x0	RD_DATA_XFER: 0 = READ 1 = NO_READ
5	0x0	ADDR_XFER_TWAIT: 0 = WAIT 1 = NO_WAIT

Bit	Reset	Description
4	0x0	ADDR_XFER_MODE: 0 = EN_8BIT 1 = EN_16BIT
3	0x0	ADDR_XFER: 0 = SEND 1 = N0_SEND
2	0x0	CMD_XFER_TWAIT: 0 = WAIT 1 = NO_WAIT
1	0x0	CMD_XFER_WDATA: 0 = WRITE 1 = NO_WRITE
0	0x0	CMD_XFER: 0 = SEND 1 = NO_SEND

36.3.30 OWR_CMD_XFER_TCTL_0

OWR COMMAND TRANSFER TIMING CONTROL Register

Offset: 078h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TWAIT

36.3.31 OWR_ADDR_XFER_TCTL_0

OWR ADDRESS TRANSFER TIMING CONTROL Register

Offset: 07ch | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TWAIT

36.3.32 OWR_DATA_XFER_TCTL_0

OWR DATA TRANSFER TIMING CONTROL Register

Offset: 080h | Read/Write: R/W | Reset: 0b00000000000000000000000000000000

Bit	Reset	Description
31:0	0x0	TWAIT

[THIS PAGE INTENTIONALLY LEFT BLANK]

37.0 PWFM CONTROLLER

The Pulse Width Frequency Modulator (PWFM) 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 PWFM 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 then divided by 256, before subdividing it further based on the programmable value locally.

There is an APB interface which interfaces the register logic to the APB bus.

The Tegra® 3 PWFM 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.

37.1 Functionality

There are four PWFM controllers on four individual pins on the chip.

Pulse Width [23:16] determines the output pulse width and must be programmed appropriately. Frequency Divider [12:0] determines the divided clock.

The output is generated from the dividers, with each divider generating one output.

37.2 PWFM Registers

37.2.1 PWM_CONTROLLER_PWM_CSR_0_0

PWM Output-0 Configuration Control Register

Offset: 000h | Read/Write: R/W | Reset: 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.

37.2.2 PWM_CONTROLLER_PWM_CSR_1_0

PWM Output-1 Configuration Control Register

Offset: 010h | Read/Write: R/W | Reset: 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.

37.2.3 PWM_CONTROLLER_PWM_CSR_2_0

PWM Output-2 Configuration Control Register

Offset: 020h | Read/Write: R/W | Reset: 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

37.2.4 PWM_CONTROLLER_PWM_CSR_3_0

PWM Output-3 Configuration Control Register

Offset: 030h | Read/Write: R/W | Reset: 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.

38.0 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 an interrupt to SW to lower temperature via DVFS, on reaching a certain thermal/voltage threshold
- Generates a signal to the CAR to reduce CPU frequency by half, on reaching a certain thermal/voltage threshold
- Generates a signal to the PMC when the temperature reaches dangerously high levels to reset the chip and sets a flag in the PMC

38.1 TSensor Registers

38.1.1 TSENSOR_COMMON_STATUS_0

Offset: 0x000 | Read/Write: R/W | Reset: 0b0

Bit	Reset	Description
0	DISABLE	WR_PROTECT_EN: When enabled no longer can values for N, TH2. TH3, SENSOR_STOP HW_FREQ_DIV_EN, THERMAL_RST_EN, TS_ADJ be changed. This bit should be written only when SW is sure of the above stated values 0 = DISABLE 1 = ENABLE

38.1.2 TSENSOR_SENSOR0_CONFIG0_0

Offset: 0x040 | Read/Write: R/W | Reset: 0b00000000000000000000000000000001

Bit	Reset	Description
31:24	0x0	N: Number of reference clock counts for which the counter runs
23:8	0x0	M: Number of reference clock pulses after which every temperature/voltage measurement is made.
7	TS_OSC_OUT	DEBUG_SEL: selects between TS and VS output to the debug bus 0 = TS_OSC_OUT 1 = VS_OSC_OUT
6	DISABLE	INTR_THERMAL_RST_EN: Enable interrupt when transitioning to level3 (.i.e thermal reset) 0 = DISABLE 1 = ENABLE
5	DISABLE	INTR_HW_FREQ_DIV_EN: Enable interrupt when transitioning to level2 (.i.e. clk slowdown) 0 = DISABLE 1 = ENABLE
4	DISABLE	INTR_OVERFLOW_EN: Enable interrupt if there is an overflow detected 0 = DISABLE 1 = ENABLE
3	DISABLE	DVFS_EN: Enable interrupts for SW based interrupts for level1 breach. 0 = DISABLE 1 = ENABLE
2	DISABLE	THERMAL_RST_EN: Enable HW based thermal reset 0 = DISABLE 1 = ENABLE
1	DISABLE	HW_FREQ_DIV_EN: Enable HW based frequency reduction 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
0	ENABLE	SENSOR_STOP: Disables the sensor from taking any reading. Set this bit to 1 to stop the sensor from taking measurements, set to 0 to start taking measurements 0 = DISABLE 1 = ENABLE

38.1.3 TSENSOR_SENSOR0_CONFIG0_VS_0

Offset: 0x044 | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
3	DISABLE	INTR_OVERFLOW_EN: Enable interrupt if there is an overflow detected 0 = DISABLE 1 = ENABLE
2	DISABLE	DVFS_EN: Enable interrupts for SW based interrupts for level1 breach. 0 = DISABLE 1 = ENABLE

38.1.4 TSENSOR_SENSOR0_CONFIG1_0

TEMPERATURE_REF Configuration Register

Offset: 0x048 | Read/Write: R/W | Reset: 0b0000000000000000xxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	0x0	TH2: Counter reference value for temperature threshold 2. Note this value can be written as long as till the Wr_En_ bit is active.
15:0	X	TH1: Counter reference value for temperature threshold 1.

38.1.5 TSENSOR_SENSOR0_CONFIG2_0

Offset: 0x04c | Read/Write: R/W | Reset: 0b0000000000000000xxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	0x0	TH0: Difference between TH1 and a safe level when we can restore the normal operation of the chip.
15:0	X	TH3: Counter reference value for temperature threshold 3. Note this value can be written as long as till the Wr_En_ bit is active.

38.1.6 TSENSOR_SENSOR0_CONFIG1_VS_0

VOLTAGE_REF Configuration Register

Offset: 0x050 | Read/Write: R/W | Reset: 0b0000000000000000xxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	0x0	TH2: Counter reference value for voltage threshold 2. Note this value can be written as long as till the Wr_En_ bit is active.
15:0	X	TH1: Counter reference value for voltage threshold 1.

38.1.7 TSENSOR_SENSOR0_CONFIG3_0

Offset: 0x054 | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	TS_ADJ: Reserved bit

38.1.8 TSENSOR_SENSOR0_STATUS0_0

Offset: 0x058 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxx0xxxxxxxx

Bit	R/W	Reset	Description
22:20	RO	X	VS_PREV_STATE: previous STATE for voltage sensor 0 = INVALID 1 = LEVEL0 2 = LEVEL1 3 = LEVEL2 4 = OVERFLOW
18:16	RO	X	VS_STATE: CURRENT STATE for voltage sensor 0 = INVALID 1 = LEVEL0 2 = LEVEL1 3 = LEVEL2 4 = OVERFLOW
10	RO	X	AVG_MIN_MAX_VALID: Valid signal for avg_count, min_count, max_count. Look at status1 and status2 registers for avg, min, max values. 0 = INVALID 1 = VALID
9	RO	X	CURRENT_VALID: Valid signal for the CURRENT_COUNT value. 0 = INVALID 1 = VALID
8	RW	CLEAR	INTR: Status of Interrupt bit. A write to this bit will clear the interrupt. This interrupt is raised when transitioning from one level to the next. 0 = CLEAR 1 = SET
6:4	RO	X	PREV_STATE: Previous STATE 0 = INVALID 1 = LEVEL0 2 = LEVEL1 3 = LEVEL2 4 = LEVEL3 5 = OVERFLOW
2:0	RO	X	STATE: CURRENT STATE for thermal sensor 0 = INVALID 1 = LEVEL0 2 = LEVEL1 3 = LEVEL2 4 = LEVEL3 5 = OVERFLOW

38.1.9 TSENSOR_SENSOR0_TS_STATUS1_0

Offset: 0x05c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	CURRENT_COUNT: Current value of the counter for the sensor (note 0xFFFF will signify overflow). Look at CURRENT_VALID to check if this value is valid.
15:0	X	AVG_COUNT: Average counter for the past 8 values of the sensor. Look at AVG_MIN_MAX_VALID to see if this value is valid.

38.1.10 TSENSOR_SENSOR0_TS_STATUS2_0

Offset: 0x060 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	MAX_COUNT: Maximum counter value, based on the last 8 values. Look at AVG_MIN_MAX_VALID bit to see if this value is valid.
15:0	X	MIN_COUNT: Minimum counter value, based on the last 8 values. Look at AVG_MIN_MAX_VALID bit to see if this value is valid.

38.1.11 TSENSOR_SENSOR0_VS_STATUS1_0

Offset: 0x064 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	CURRENT_COUNT: Current value of the counter for voltage sensor (note 0xFFFF will signify overflow). Look at CURRENT_VALID to check if this value is valid.
15:0	X	AVG_COUNT: Average counter for the past 8 values. Look at AVG_MIN_MAX_VALID to see if this value is valid.

38.1.12 TSENSOR_SENSOR0_VS_STATUS2_0

Offset: 0x068 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	MAX_COUNT: Maximum counter value, based on the last 8 values. Look at AVG_MIN_MAX_VALID bit to see if this value is valid.
15:0	X	MIN_COUNT: Minimum counter value, based on the last 8 values. Look at AVG_MIN_MAX_VALID bit to see if this value is valid.

38.1.13 TSENSOR_SENSOR1_CONFIG0_0

Offset: 0x080 | Read/Write: R/W | Reset: 0b00000000000000000000000000000001

Bit	Reset	Description
31:24	0x0	N: Number of ref clk counts for which the counter runs
23:8	0x0	M: Number of reference clock pulses after which every temperature/voltage measurement is made.
7	TS_OSC_OUT	DEBUG_SEL: selects between TS and VS output to the debug bus 0 = TS_OSC_OUT 1 = VS_OSC_OUT
6	DISABLE	INTR_THERMAL_RST_EN: Enable interrupt when transitioning to level3 (.i.e thermal reset) 0 = DISABLE 1 = ENABLE
5	DISABLE	INTR_HW_FREQ_DIV_EN: Enable interrupt when transitioning to level2 (.i.e. clk slowdown) 0 = DISABLE 1 = ENABLE
4	DISABLE	INTR_OVERFLOW_EN: Enable interrupt if there is an overflow detected 0 = DISABLE 1 = ENABLE
3	DISABLE	DVFS_EN: Enable interrupts for SW based interrupts for level1 breach. 0 = DISABLE 1 = ENABLE

Bit	Reset	Description
2	DISABLE	THERMAL_RST_EN: Enable HW based thermal reset 0 = DISABLE 1 = ENABLE
1	DISABLE	HW_FREQ_DIV_EN: Enable HW based frequency reduction 0 = DISABLE 1 = ENABLE
0	ENABLE	SENSOR_STOP: Disables the sensor from taking any reading. Set this bit to 1 to stop the sensor from taking measurements, set to 0 to start taking measurements 0 = DISABLE 1 = ENABLE

38.1.14 TSENSOR_SENSOR1_CONFIG0_VS_0

Offset: 0x084 | Read/Write: R/W | Reset: 0b00

Bit	Reset	Description
3	DISABLE	INTR_OVERFLOW_EN: Enable interrupt if there is an overflow detected 0 = DISABLE 1 = ENABLE
2	DISABLE	DVFS_EN: Enable interrupts for SW based interrupts for level1 breach. 0 = DISABLE 1 = ENABLE

38.1.15 TSENSOR_SENSOR1_CONFIG1_0

TEMPERATURE_REF Configuration Register

Offset: 0x088 | Read/Write: R/W | Reset: 0b0000000000000000xxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	0x0	TH2: Counter reference value for temperature threshold 2. Note this value can be written as long as till the Wr_En_ bit is active.
15:0	X	TH1: Counter reference value for temperature threshold 1.

38.1.16 TSENSOR_SENSOR1_CONFIG2_0

Offset: 0x08c | Read/Write: R/W | Reset: 0b0000000000000000xxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	0x0	TH0: Difference between TH1 and a safe level when we can restore the normal operation of the chip.
15:0	X	TH3: Counter reference value for temperature threshold 3. Note this value can be written as long as till the Wr_En_ bit is active.

38.1.17 TSENSOR_SENSOR1_CONFIG1_VS_0

VOLTAGE_REF Configuration Register

Offset: 0x090 | Read/Write: R/W | Reset: 0b0000000000000000xxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	0x0	TH2: Counter reference value for voltage threshold 2. Note this value can be written as long as till the Wr_En_ bit is active.
15:0	X	TH1: Counter reference value for voltage threshold 1.

38.1.18 TSENSOR_SENSOR1_CONFIG3_0

Offset: 0x094 | Read/Write: R/W | Reset: 0b00000000

Bit	Reset	Description
7:0	0x0	TS_ADJ: Reserved bit

38.1.19 TSENSOR_SENSOR1_STATUS0_0

Offset: 0x098 | Read/Write: R/W | Reset: 0bxxxxxxxxxxxxxxx0xxxxxxxx

Bit	R/W	Reset	Description
22:20	RO	X	VS_PREV_STATE: previous STATE for voltage sensor 0 = INVALID 1 = LEVEL0 2 = LEVEL1 3 = LEVEL2 4 = OVERFLOW
18:16	RO	X	VS_STATE: CURRENT STATE for voltage sensor 0 = INVALID 1 = LEVEL0 2 = LEVEL1 3 = LEVEL2 4 = OVERFLOW
10	RO	X	AVG_MIN_MAX_VALID: Valid signal for avg_count, min_count, max_count. Look at status1 and status2 registers for avg, min, max values. 0 = INVALID 1 = VALID
9	RO	X	CURRENT_VALID: Valid signal for the CURRENT_COUNT value. 0 = INVALID 1 = VALID
8	RW	CLEAR	INTR: Status of Interrupt bit. A write to this bit will clear the interrupt. This interrupt is raised when transitioning from one level to the next. 0 = CLEAR 1 = SET
6:4	RO	X	PREV_STATE: Previous STATE 0 = INVALID 1 = LEVEL0 2 = LEVEL1 3 = LEVEL2 4 = LEVEL3 5 = OVERFLOW
2:0	RO	X	STATE: CURRENT STATE for thermal sensor 0 = INVALID 1 = LEVEL0 2 = LEVEL1 3 = LEVEL2 4 = LEVEL3 5 = OVERFLOW

38.1.20 TSENSOR_SENSOR1_TS_STATUS1_0

Offset: 0x09c | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	CURRENT_COUNT: Current value of the counter for sensor (note 0xFFFF will signify overflow). Look at CURRENT_VALID to check if this value is valid.
15:0	X	AVG_COUNT: Average counter for the past 8 values of the sensor. Look at AVG_MIN_MAX_VALID to see if this value is valid.

38.1.21 TSENSOR_SENSOR1_TS_STATUS2_0

Offset: 0x0a0 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	MAX_COUNT: Maximum counter value, based on the last 8 values. Look at AVG_MIN_MAX_VALID bit to see if this value is valid.
15:0	X	MIN_COUNT: Minimum counter value, based on the last 8 values. Look at AVG_MIN_MAX_VALID bit to see if this value is valid.

38.1.22 TSENSOR_SENSOR1_VS_STATUS1_0

Offset: 0x0a4 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	CURRENT_COUNT: Current value of the counter for voltage sensor (note 0xFFFF will signify overflow). Look at CURRENT_VALID to check if this value is valid.
15:0	X	AVG_COUNT: Average counter for the past 8 values. Look at AVG_MIN_MAX_VALID to see if this value is valid.

38.1.23 TSENSOR_SENSOR1_VS_STATUS2_0

Offset: 0x0a8 | Read/Write: RO | Reset: 0bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Bit	Reset	Description
31:16	X	MAX_COUNT: Maximum counter value, based on the last 8 values. Look at AVG_MIN_MAX_VALID bit to see if this value is valid.
15:0	X	MIN_COUNT: Minimum counter value, based on the last 8 values. Look at AVG_MIN_MAX_VALID bit to see if this value is valid.

[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